Software Engineering Institute

COTS and Reusable Software Management Planning: A Template for Life-Cycle Management

William Anderson
Ed Morris
Dennis Smith
Mary Catherine Ward

October 2007

TECHNICAL REPORT
CMU/SEI-2007-TR-011
ESC-TR-2007-011

Acquisition Support Program
Dynamic Systems Program

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgments

Development of this template has been a collaboration of many people dedicated to establishing the best possible software management strategies. We would like to particularly acknowledge the following contributors: Michele Shaw of Fraunhofer Center and Marilyn Goo, Sue Hermanson, and Samantha Montgomery of Boeing Company

Abstract

The acquisition community needs guidance in long-term management planning for selecting, approving, and upgrading software products, especially commercial off-the-shelf (COTS) and other reusable software products. As the mixture of these components in systems increases, the demand for a planned way to manage them continues to grow.

The COTS and Reusable Software Management Plan (CRSMP) can facilitate acquisition programs' management of COTS and other reusable software products. The CRSMP provides a strategy outline for managing data about component licensing, tracking release schedules, monitoring software interdependencies, choosing specific features and extensions and documenting those choices, and evaluating and mitigating risks associated with deploying COTS and other reusable software components in a system.

The CRSMP presented in this report can serve as a guide for how to manage multiple COTS and other reusable software components in complex systems.

1 Introduction

The acquisition community needs guidance in long-term management planning for selecting, approving, and upgrading software products, especially commercial-off-the-shelf (COTS) and reusable software products. Several program offices have identified the need to manage their systems' components in a more methodical way. As the mixture of these components in systems increases, the demand for a planned way to manage them continues to grow. Not planning and controlling the use of COTS and other reusable software components throughout a system's life cycle could increase system costs, speed obsolescence, or render it inoperable.

The Acquisition Support Program from the Carnegie Mellon University® Software Engineering Institute (SEI) recently sponsored a pilot to develop a COTS software management plan. The SEI's knowledge of COTS-based systems combined with the expertise of a large government program office provided a solid foundation for the development of this plan. But because COTS software is just one type of reusable software component, the scope of the project was expanded to be more comprehensive. The result is this COTS and Reusable Software Management Plan (CRSMP) template.

The CRSMP outlines a strategy for selecting, approving, and upgrading common reusable software components. It includes techniques for managing component licensing data, tracking release schedules, monitoring software interdependencies, choosing specific features and extensions and documenting those choices, and evaluating and mitigating risks associated with deploying COTS and other reusable software components in a system.

The CRSMP contained in this report is formatted as a template. You must customize the different sections and the content within the sections for use in your organization's acquisition environment. After you have created a customized plan, it can serve as a guide for how to manage multiple COTS and other reusable software products in complex systems. The icons used in this document denote the following:

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

The folder icon denotes a summary of the content that is included in a section.

The question mark icon denotes a list of questions to consider when adding information that is specific your organization's acquisition environment. The lists of questions included in the template are not exhaustive; they are only meant to generate ideas about what information you should include.

Each section of the template contains sample text from an actual CRSMP. The sample text has been sanitized, is prefaced with the title "Sample Text," and is differentiated from other text in the report by gray background shading. The text in these examples is not meant to be used in your organization's CRSMP.

2 Preliminary Sections

The sections listed in Section 2 are preliminary sections that you may want to include in your organization's CRSMP. For example, you can choose to include an abstract to introduce the purpose of your CRSMP document to those who may be unfamiliar with it.

2.1 ACKNOWLEDGEMENTS

Developing a CRSMP requires input from many people who represent many different groups and organizations. It is appropriate to recognize them for their time and effort.

Sample Text

See the section titled "Acknowledgements" in this document for an example.

2.2 ABSTRACT

An abstract provides a description of the document that is concise and self contained. It describes the context and content of the document in 150 words or less.

Sample Text

See the section titled "Abstract" in this document for an example.

2.3 TABLE OF OPEN ITEMS

This section is a placeholder for a table that will allow you to track open items and issues.

3 Scope

Establish the boundaries of the CRSMP.

- What is the scope of the CRSMP?
- What organizations are subject to the plan?
- What types of reusable software components apply to the program and how are they obtained?

Sample Text

This CRSMP applies to organizations that select, evaluate, and use reusable software components. It includes techniques for managing component licensing data, tracking release schedules, monitoring software interdependencies, choosing specific features and extensions and documenting those choices, and evaluating and mitigating risks associated with deploying COTS and other reusable software components in a system.

Reusable software components can be grouped into categories of components that are obtained from different sources. The following list includes descriptions of several kinds of reusable software components and their sources:

- Commercial off-the-shelf (COTS)–Any software, hardware, or service item that is offered for sale, lease, license, or free of charge to the general public in multiple, identical copies and used without modification of the internals. COTS software products are supported and evolved by the vendor, who retains the intellectual property rights. Source code for COTS software is generally not available to the implementation team [Comella-Dorda 2003].
- Government off-the-shelf (GOTS)–GOTS software products have characteristics that are similar to those of a COTS product, except that they are developed and owned by the government, may be available to a restricted customer base, may provide source code, and may have varying levels of support and evolution from the government.
- Shareware–Software that can be obtained and redistributed for free, but most often is under copyright and does legally require a payment, at least when it is used beyond the evaluation period or for commercial applications. Sometimes this is a fully featured product; other times

it lacks some of the features of the commercial version. If you find the product useful, you are expected to register the software.

- Freeware–Copyrighted software that is made available without charge. In contrast to shareware, unlimited personal usage is permitted, but you cannot do anything else without express permission of the creator. The program may not be resold or distributed by others for profit.
- Open Source–Software source code that is available to the general public and that does not have licensing restrictions that limit use, modification, or redistribution. The Open Source Initiative reviews and certifies open source programs. Among the initiative's stringent criteria is the requirement that no one collect a royalty on the software and no person, group, or field of endeavor be denied access to the program.
- Supplier-Sourced Software–Software suggested and offered by the supplier for inclusion in the system. This software may be made available in manners similar to COTS, GOTS, Shareware, Freeware, and Open Source, or in some combination of their characteristics. In general, however, it is not true COTS software because it is not licensed to a large, diverse, and public user base, and the supplier may expect to modify the software for specific purposes.
- Other Non-Developmental Item (NDI) software–A catch-all category that includes any previously developed software item used exclusively for government purposes by a federal agency, a state or local government, or a foreign government with which the United States has a mutual defense cooperation agreement.

While these categories are useful for grouping different kinds of reusable software components, they are too general to be useful for decision making. First, the labels used for the different categories ("COTS," "GOTS," "shareware," etc.) can encompass extremely divergent software products. For example, the COTS software category can include both products developed by well-known software companies like Oracle and those created by small, relatively unknown "start-up" companies. It is very unlikely that the products produced by the well-known companies would have much in common with the start-up companies with regard to vendor capability and customer support.

Also, many reusable software components possess unique characteristics that make associating them with a single category difficult. For example, a vendor of a COTS product that is offered for use by the general public may create and maintain a "militarized" version. While the product retains some characteristics of a COTS software component, some people would categorize it as "non-COTS" because the software may no longer be marketed to the general public and maintenance is not performed in the way it is for commercially available products.

Reusable software components should not be evaluated and managed in a manner based on broad categorizations (e.g., COTS or GOTS). Instead, the characteristics that a component possesses provide a more suitable basis for evaluation and life-cycle planning and maintenance.

3.1 IDENTIFICATION

This section contains the program-specific identification of this CRSMP document within your program's documentation structure. For example, contract data requirements list (CDRL) number, specific requirement met, document number, and so on.

3.2 PURPOSE

Describe the organization's motivation for reusing software components and for creating a CRSMP to help manage them.

- What is your organization's motivation for using COTS and other reusable software components?
- Are there cost, schedule, or technical motivations? If so, describe.
- Are there contractual mandates?
- Why does your organization need to manage COTS and other reusable software components?
- What key processes are unique to this CRSMP? Do any of them help address the risks associated with the development of a system that uses a combination of reusable software components?

Sample Text

In order to meet aggressive delivery schedules, the program uses COTS and other reusable software components as appropriate.

Use of COTS and other reusable software components has its challenges. While the sources of these types of software vary, they have two key characteristics in common from the perspective of an organization attempting to use them: imprecise knowledge of the internals (e.g., architecture,

design, assumptions, and dependencies) and limited control over the evolution of the component.

It can be extremely difficult to integrate independently developed components, and to sustain integration across frequent and uncontrollable software releases. Often, problems experienced can be directly traced to imprecise knowledge and limited control. These problems are manifest in faulty selection processes, conflicts between components, inappropriate integration strategies, and an inability to sustain the component across the system life cycle. Careful planning of the processes, techniques, and artifacts associated with reusable software components can help organizations avoid or overcome common problems due to imprecise understanding and to prepare for both expected and unexpected situations.

The purpose of this CRSMP is to identify collaborative processes and techniques use to manage reusable software components and to address issues and risks. The following actions are among those that should be taken to address risks and issues:

- Establish a COTS and other reusable software component usage tracking system (known as the Software Reuse Item Database (SRIDB) including component information from COTS vendors who often incorporate other COTS components into their products.
- Establish a COTS and other reusable software component evaluation process that addresses commonality, interoperability, and other criteria established in the Component Evaluation Criteria (CEC) and documented in the program's Software Development Plan (SDP).
- Identify components that can be shared across the system and encourage their use.
- Eliminate unjustified proliferation of similar but different reusable software components across the organization and support the negotiation of quantity discounts where possible.
- Share COTS and other reusable software components' evaluation information, including benchmarking data, across the organization.
- Enable reusable software component evolution that is consistent with a spiral development model.
- Coordinate requests for government approval for delivery with special licensing rights.

3.3 PROGRAM OVERVIEW

Describe the specific program that the CRSMP is to guide.

- What is the scope of the program?
- What is the operational impact of the system?
- Who are the primary stakeholders?
- What aspects of the program and system context are relevant to the management of COTS and other reusable software components?

3.4 DEFINITIONS

Describe program-specific definitions specific to this CRSMP in this section.

- How will unique terminology be handled in the CRSMP?
- Will you use a special formatting to denote it?

3.5 DOCUMENT OVERVIEW

Describe how the CRSMP is organized.

Sample Text

This CRSMP includes the following sections:

- Scope–Provides introductory information concerning the CRSMP for the program and describes its relationship to other program plans.
- Applicable Documents–Lists government, compliance, and other reference documents pertinent to the CRSMP.
- Strategies for Managing and Developing COTS and Other Reusable Software Components–Describes the program's strategies and approaches for managing and developing COTS and other reusable software components, which provide the basis for the supporting roles and processes defined in this CRSMP.
- Roles, Responsibilities, and Relationships–Provides a summary of the program roles affected by this plan and includes an overview of the concept of operations that support the CRSMP strategies.
- Process Artifacts–Describes the key artifacts that will be developed in accordance with the CRSMP.
- Process Descriptions–Describes the program's processes that ensure implementation of the CRSMP strategies. Each process description includes a summary of the process, description of the artifacts, and associated roles and responsibilities.
- Managing the Life Cycles of COTS and Other Reusable Software Components– Describes the activities that transcend the development of systems that employ reusable software components. The activities described in this section are executed for the entire life cycle of the system.
- Appendix A EPIC Overview–Provides an overview of the Evolutionary Process for Integrating COTS-based systems (EPIC[SM]). EPIC defines acquisition, management, and engineering practices that effectively leverage the COTS marketplace and other sources of pre-existing components.
- Appendix B OAR Overview–Provides an overview of the Options Analysis for Reengineering (OAR[SM]) method. OAR is a systematic, architecture-centric approach for identifying and mining reusable software components within large, complex, software systems.
- Appendix C PECA Overview–Provides an overview of the Plan, Establish, Collect, Analyze

[SM] EPIC and OAR are service marks of the Carnegie Mellon Software Engineering Institute.

(PECA) evaluation process. This process helps organizations make carefully reasoned decisions when choosing COTS and other reusable software components.

- Appendix D Guidelines for Generation of Evaluation Criteria–Provides suggestions for generating broad-ranging criteria for evaluating COTS and other reusable software components. The appendix includes a taxonomy that can serve as a starting point, but does not provide actual criteria.

- Appendix E Component Evaluation Record–Provides suggestions for documenting the outcome of an evaluation activity.
- Appendix F Reuse Evaluation Report–Provides suggestions for documenting the analysis process used to compare the characteristics of potentially reusable software components.
- Appendix G Information Needed for Market Watch–Describes the information needed to track market segments that produce reusable software components.

3.6 RELATIONSHIPS TO OTHER PLANS

Clarify the relationships that this document has to other relevant program-specific documentation.

- What other plans have an impact on this CRSMP?
- Does this plan contain contractually binding direction, or does it provide guidance for developing processes that meet contractual stipulations that are specified elsewhere?
- Is a precedent order necessary? Who resolves the conflicts between plans?

Sample Text

This plan implements the COTS and other reusable software component strategies required by the SDP. Other plans that affect this CRSMP include the following:

- Software Configuration Management Plan (SCMP)–The SCMP governs the configuration of all software including COTS and other reusable software components.
- Software Management Plan (SMP)–Data regarding COTS and other reusable software components are collected as defined in the SMP.

- Software Quality Assurance Plan (SQAP)–The SQAP defines the quality activities for the program including COTS and other reusable software components.
- Phased Software Integration Plan (PSIP)–The PSIP defines the strategies for component integration including COTS and other reusable software components. To the extent practical, upgrading to new releases of COTS and other reusable software products is included in the PSIP.
- Software Transition Plan (STrP)–The STrP addresses software transition issues for COTS and other reusable software components.
- Software Risk Management Plan (SRMP)–The SRMP defines strategies for identifying, managing, and mitigating risks associated with developed and reusable software components.

4 Applicable Documents

List applicable documents.

4.1 GOVERNMENT DOCUMENTS

List applicable government documents.

4.1.1 Reference Documents

List applicable government reference documents.

4.1.2 Compliance Documents

List government compliance documents.

4.2 CONTRACTOR DOCUMENTS

List applicable contractor documents.

4.2.1 Contractor Reference Documents

List contractor reference documents.

4.2.2 Contractor Compliance Documents

List contractor compliance documents.

4.3 OTHER REFERENCE DOCUMENTS

List other reference documents.

Sample Text

The following documents were referenced in the development of this CRSMP:

1. Albert, C. & Brownsword, L. *Evolutionary Process for Integrating COTS-Based Systems (EPIC): Building, Fielding, and Supporting Commercial off-the-Shelf (COTS) Intensive Solutions* (CMU/SEI-2002-TR-005). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.
2. Albert, C. & Brownsword, L. *Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview* (CMU/SEI-2002-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.
3. Bergey, J.; O'Brien, L.; & Smith, D. *Options Analysis for Reengineering (OAR); A Method for Mining Legacy Assets* (CMU/SEI-2001-TN-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001.
4. Clements, P. & Northrop, L. *Software Product Lines: Practices and Patterns*. New York, NY: Addison-Wesley, 2001.
5. Comella-Dorda, S.; Dean, J.; Lewis, G.; Morris, E.; Oberndorf, P.; & Harper, E. *A Process for COTS Software Product Evaluation* (CMU-SEI-2003-TR-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
6. *Standard for Information Technology, Software Life Cycle Processes, Software Development, Acquirer-Supplier Agreement (Interim Standard–J-STD-016)*. Electronics Industries Association (EIA)/Institute of Electrical and Electronics Engineers (IEEE), September 1995.
7. Saaty, T.L. *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill, 1980.

5 Strategies for Managing and Developing COTS and Other Reusable Software Components

Describe the high-level, significant strategies that will affect the program's use of reusable software components.

Note: The subsections you include in this section depend on the specific strategies selected by your program.

- What are the reusable software component strategies for your program?
- What are the other key strategies for the program?
- What are the drivers behind these strategies?

Sample Text

This program faces several challenges:

- Deliver a new kind of system that provides unprecedented capability to users.
- Begin constructing while requirements are not well understood.
- Meet aggressive delivery schedules.

To meet these challenges, the program employs strategies that incrementally refine requirements and aggressively reuse software components. These strategies include

- development strategies
- product line strategies
- program-wide component reuse strategies
- creation of a SRIDB

5.1 DEVELOPMENT STRATEGIES

Describe the software development strategies that are related to the CRSMP.

- What type of software development model (e.g., spiral, iterative, rapid application development, waterfall, agile) is used for the program?
- How can we describe the development model at a high level in this section and still provide guidance specific to COTS and other reusable software components in later sections?
- Can we represent the model graphically on a single page?
- Can we provide basic guidance for tailoring the selected model?

Sample Text

Development of COTS and other reusable software components does not lend itself to the waterfall development model, since neither the reusable software components nor our understanding of them are static. Managing the dynamic nature of reusable software components, their interactions with each other, and the rest of the system is critical to effective reuse. The processes identified

within a CRSMP must support mechanisms for reconsideration and re-execution of steps as we learn more about components through better understanding and risk-reduction activities.

The program has adopted a WinWin Spiral approach [USC 2002]. The spiral model is a risk-driven approach that develops software and system requirements, and architectural solutions as win conditions negotiated among a project's stakeholders. The spiral model addresses the flaws in traditional approaches that require complete knowledge of a system's requirements and architecture before the development cycle. The spiral model is particularly appropriate for large-scale, cutting edge development efforts such as this program, where the premise of complete system knowledge is unquestionably false.

This CRSMP is consistent with the WinWin Spiral model and directs that the software teams involved with a development effort perform the following tasks:

- Apply a spiral model internally to develop system requirements and architectures that represent common agreement between developers, customers, and other stakeholders.
- Provide early notification of externally significant alternatives, problems and risks, and decisions that potentially affect other software teams.
- Track risks, risk mitigation activities, and decisions made regarding internally and externally significant concerns.

The application of the spiral model to COTS and other reusable software components means that as additional information becomes available, preliminary decisions regarding system objects, architecture, and component selection may be reconsidered and associated artifacts updated. During initial builds, COTS and other reusable software components are selected and evaluated. In later builds, the components are upgraded to new versions. New COTS and other reusable software components are discovered and undergo the evaluation and approval process. System evolution follows the same phases of the spiral model as custom software: inception, elaboration, construction, and transition.

Section 8, Process Descriptions, describes the processes and associated artifacts necessary for managing complex, high-risk COTS and other reusable software components having a large number of stakeholders. The supplier or software team tailors these processes and the formality of artifacts depending on the particular component's characteristics. The supplier or software team needs to understand the characteristics of the component to tailor the processes and artifacts. The following factors can influence the degree to which processes and artifacts are modified:

- The component represents a low risk to program, or has low involvement with program strategic objectives.
- The impact of the component is effectively isolated within a part of the system.
- There are readily available and cost-effective alternatives should a component fail to meet expectations.
- Specific characteristics about the component or its environment dominate other characteristics, thereby invalidating baseline expectations.
- A component represents such a dominant solution that it renders the normal evaluation process meaningless.
- Extensive experience with a component allows compression of some processes.
- The required source, size, complexity, and pedigree of the component limit available choices.

The range of potential modifications to the baseline COTS and other reusable software component management processes is large. For low-risk components, suppliers and software teams can modify the following:

- formality of various plans (for example, the evaluation plan or the life-cycle management plan)
- rigor of the evaluation and other processes
- documentation requirements

Reductions in the scope of plans, processes, and documentation require approval based on the impact of the component being considered. For components with no effect outside of a supplier, the supervising software team will hold approval authority. For components with impact limited to a module, the module's architect will hold approval authority. For components that have effects across multiple modules, the program system architect will hold approval authority for all changes to expectations regarding plans, processes, and documentation. Further information for determining whether a component has impact only for the supplier, for a module, or for multiple modules can be found in Section 8.1, Inception.

At minimum, we will perform the following steps:

1. categorize components, including approval
2. create an abbreviated Component Evaluation Record (CER)
3. apply evaluation criteria

4. create a Reuse Evaluation Analysis Report
5. gather support data for transition

It is important to recognize that use of the spiral model implies that many activities that occur during the inception, elaboration, construction, and transition phases occur multiple times before a release, as depicted in Figure 1.

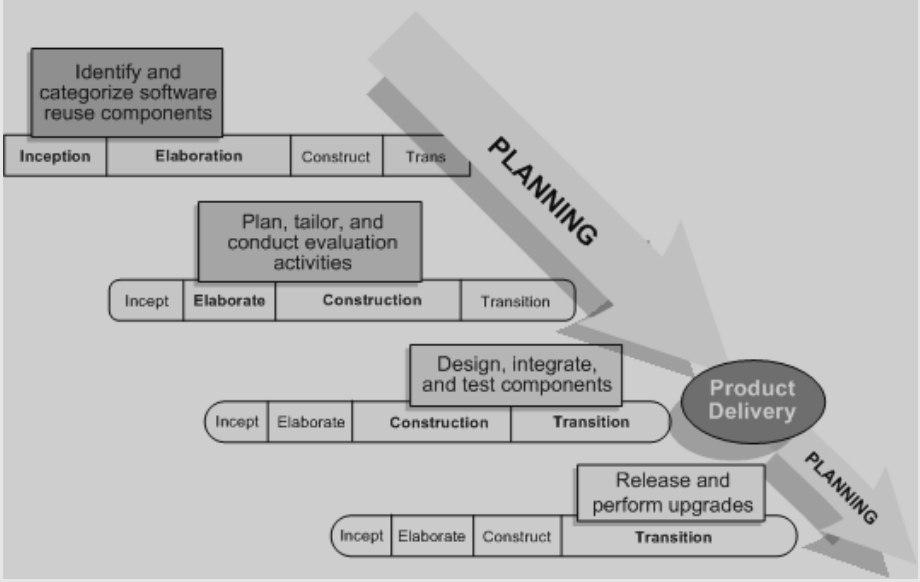

Figure 1: Managing COTS and Other Reusable Software Using the Spiral Model

In addition, software teams and their suppliers execute different phases of the model at different times. Thus, one software team may have progressed to the point of transitioning a robust initial capability, while another software team may be performing risk reduction activities to assist in the identification of baseline requirements.

It is the responsibility of all members of the team to make this possible by performing the following tasks:

- Identify components that potentially have impact outside of the immediate scope.
- Elevate component decisions to the appropriate level.
- Inform others of progress toward component selection and integration.

- Analyze the component selections of others for potential impact.
- Negotiate disagreements in good faith and with the common good of the program in mind.
- Identify places where standards are needed and follow standards where identified.
- Document component expectations and use.
- Use the risk-reporting and tracking mechanisms to inform the community of potential problems.

This document does not provide further definition of the spiral model, except to provide specific COTS and other reusable software component guidance regarding major activities that occur at various phases of the process. This does not preclude the same activities from occurring at other phases of the WinWin Spiral model, but simply places them at the point of most common occurrence.

5.2 PRODUCT LINE DEVELOPMENT STRATEGIES

Define the component-based development approach used for your program.

- Will you be using a software product line approach or some other approach that encourages reuse across a range of systems?

Sample Text

In order to maximize reuse of common architectures, designs, and implementations across closely related systems, a product line approach will be employed where appropriate.

5.2.1 Using Product Lines

Determine whether your program uses a product line or other component-based development approach.

- What is a software product line?
- How can you determine if using a software product line benefits your organization?
- Will your organization be using a software product line approach? If so, how? If not, why not?
- How will you manage the reuse of previously existing software components such as COTS software, legacy software, or non-developmental software components?

Sample Text

A *software product line* is "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market module or mission and that are developed from a common set of core assets in a prescribed way." [SEI 2007] The reusable core assets include software components, but also the associated documentation, architecture, software design, application programming interface (API), user interface design, test plans, test cases, schedules and budgets, development processes, and more. Building a set of software systems as a software product line has been shown to dramatically shorten development time, increase productivity, increase quality, and reduce cost, as compared to developing the systems one at a time in isolation.

A primary difference between a software product line approach to reuse and standard software reuse techniques is that the core asset base is maintained so that it is always applicable to all of the products it serves. In standard reuse, an asset is discovered, modified as necessary, and installed in the product. While this saves some development time, this "clone-and-own" method makes each product unique, with maintenance and evolution no longer shared with other members of the family. In a software product line, the core asset is built to be used as is (or instantiated using preplanned variation mechanisms such as parameterization or configuration files) in all products for which it was developed. As the set of products grows and evolves, the core asset base is updated so that all products continue to be buildable derivatives of those assets. When an error

is discovered and fixed in a core asset, or when a core asset is updated to provide enhanced capability, all products using that core asset benefit.

Programs should determine if their systems lend themselves to the software product line approach. For example, the software that runs on all vehicles of a major manufacturer (car, truck, or van) may constitute a software product line. As another example, a training team might develop common training enabler software items that are used for embedded trainers as well as a range of stand-alone task trainers. The following section outlines how we will determine if we will use a software product line approach.

5.2.1.1 Strategies for Developing a Software Product Line

Sample Text

To effectively implement the software product line approach, each development team having software product responsibility and including suppliers, must perform the following tasks:

- Determine which systems or parts of systems are potential members of a software product line. Criteria include analyses of mission improvement, product quality goals, costs, targets for product development and delivery, and product risk reduction. Product lines need not be limited in scope to the team making the proposal; cross team product lines may also be developed. The overarching software authority identifies potential software product lines based on the descriptions of software components in the software architectures produced by lower level teams.
- Define the potential software product line's scope. This involves identifying the entities with which items/assets in the software product line will interact, describing what the items/assets in the software product line will have in common, and how the items/assets in the product line will vary from each other.
- Identify the core assets that could be used across the software product line. For each core asset, identify the products in which it could be used. For each core asset, describe the variability necessary to support the products in the software product line.
- For each core asset, perform a cost/benefit analysis to compare the cost of producing the asset with the savings afforded by using the asset.

If the reuse value of an asset is high, do the following:

- Develop a product line architecture that identifies allowable variations and the mechanisms for achieving them. Examples of some variations are: 1) including or omitting an optional

component, 2) including a variable number of components, 3) including alternate implementations of a component, or 4) configuring a component using mechanisms such as parameters, configuration files, or specializations of particular classes.

- Identify components that will populate the architecture. Determine how to acquire components with sufficient flexibility to support the variation points specified in the product line architecture. Core assets for a software product line may include new and reused existing software.
- Develop a production plan to control how core assets are to be used (how their variation mechanisms are to be exercised) and integrated into products.
- Define the (management) process for maintaining the product line's core assets and their evolution so that the core asset base remains relevant as the ongoing foundation for the products in the software product line.
- Design and code software product line core assets using standards that enhance reusability. Once tested, the core assets will be placed in a library and maintained as part of the software configuration management database, along with any build scripts, and so forth, that are used to instantiate the core assets in products.

If the reuse value of an asset is less than very high, other reuse strategies such as those identified in Section 5.3,

Program-Wide Component Reuse Strategy, may be appropriate.

5.3 PROGRAM-WIDE COMPONENT REUSE STRATEGY

Describe the program's approach to using COTS and other reusable software components.

- What are the key strategies for your program regarding COTS and other reusable software components?
- What are the drivers behind these strategies?

Sample Text

The use of COTS and other reusable software components is based on standards whenever possible. Modifying COTS software products is avoided, even when the vendor is willing to create a custom version. Modifications to other types of reusable software components and code to adapt COTS products to the program environment is carefully controlled and monitored to insure that the integrity of components is maintained and that no long-term sustainment problems are incurred.

The program strategy involves performing the following tasks:

- Apply engineering processes that achieve increasingly better understanding of system requirements, architectures, designs, and components.
- Develop strategies and construction of requirements that encourage reuse.
- Aggressively identify potential reuse components as early as possible.
- Document reuse activities to support both short- and long-term engineering and management needs and to provide testimony to a sound engineering process.
- Categorize and manage components depending on the level of reuse and the potential effect on other parts of the system.
- Rigorously evaluate the suitability of components and component providers.
- Implement processes required to adapt to new knowledge of requirements, architecture, and other aspects of the system.
- Implement processes that document notification of changes to reused components and related requirements and strategies in a timely manner.
- Consider the effect of components on the system beyond the development cycle (for the full life cycle).

The program recognizes that as new information becomes available about system requirements, architecture, and design, some reuse decisions may have to be reconsidered. This is facilitated by early recognition and sharing of information regarding the status of the system and reuse components. Managing components according to the level and impact of reuse allows the program to appropri-

ately share information between the organization and suppliers, with the goal of mitigating the risks of poor capability and performance.

Reuse component evaluations are based on rigorous processes that take into consideration the impact of a reuse system component's API, provider characteristics, component architecture, release frequency and reliability, and other factors.

Finally, the evaluation process takes into account the fact that reusable software components must be actively monitored for changes in component and provider status. Part of this process is a market-watch activity that analyzes trends that are indicative of a pending component upgrade or the release of new components.

6 Roles, Responsibilities, and Relationships

Describe the organizational roles and responsibilities with respect to managing COTS and other reusable software components.

- What is the organizational structure?
- Who is responsible for creating and maintaining key CRSMP artifacts?

Sample Text

This section describes the key roles and responsibilities for managing COTS and other reusable software components.

The program is organized such that a prime contractor, reporting to the government sponsor, is responsible for overseeing and integrating the work of several teams. Each team is focused on providing a key capability for the developing system of systems (e.g., communications infrastructure, command and control capability). A team oversees and integrates the work of several suppliers that develop portions of the key capability for which the team is responsible. The following sections define key roles, several of which exist at multiple levels within the overall program structure.

(Ideally, your CRSMP includes a high-level diagram of roles, responsibilities, and key artifacts.)

6.1 MANAGEMENT ORGANIZATION

Describe the roles and responsibilities for managing COTS and other reusable software components from both program and software management perspectives.

- What are the roles and responsibilities?
- What are the key artifacts? Who creates and maintains them?

Sample Text

The software management organization consists of the government sponsor, the program manager, program directors and managers with software responsibilities, and other organizations that support the management organization, such as those that handle contracts and procurement.

The software management organization is responsible for performing the following tasks:

- Coordinate approvals for software components that are identified as reusable.
- Communicate common reuse decisions through appropriate communication vehicles.
- Develop appropriate metrics to support decision-making processes for COTS and other reusable software components.
- Collect and analyze data about licensing, dependencies, extensions, risks, upgrade schedules, and metrics. Develop a consolidated upgrade strategy.
- Develop and maintain a database to track the use of COTS and other reusable software components across programs.
- Manage licenses for products that are used by multiple teams or assign one team to manage licenses for such products.

6.1.1 Government/Sponsor

Describe the key roles and responsibilities of the government/sponsor in managing COTS and other reusable software components.

- What are the key roles and responsibilities?

Sample Text

As the party ultimately held responsible, the government acquisition executive or designee participates in the generation of criteria for the evaluation, selection, and sustainment of COTS and other reusable software components to ensure that sound practices are followed.

This government entity, consulting with contractors as appropriate, performs the following tasks:

- Identify guidelines for software reuse within the program.
- Participate in the analysis of high-risk versus valuable software reuse decisions.
- Monitor the execution of software reuse processes and the generation of software reuse artifacts.
- Review and approve exceptions to safety, security, and data rights requirements.
- Review and approve software reuse decisions.
- Participate in reusable software component health checks.

The government acquisition executive may develop a policy for identifying categories of software reuse decisions and subsequent tailoring of software reuse decision categories, processes, artifacts, and approvals.

6.1.2 Program Manager

Describe the key roles and responsibilities of the program manager (PM) in managing COTS and other reusable software components.

- What is the key role of the PM with regard to the CRSMP?
- What are the PM's responsibilities?

Sample Text

The PM oversees all aspects of the program, including staffing, schedule, budget, and program performance for hardware, developed software, and reusable software components.

The PM's responsibilities that are related to reusable software components include the following:

- Help the sponsor and program personnel to develop appropriate strategies and guidance.
- Exchange information regarding software reuse and decisions with the sponsor.
- Provide leadership in following software reuse process guidelines, including maintaining adequate resources and staff to perform related tasks.
- Periodically review software reuse activities and factors, including strategic direction, software reuse decisions and component health, and risks and problems.

6.1.3 Software Program Director and Managers

Describe the key roles and responsibilities of the software program director and supporting managers in managing COTS and other reusable software components.

- What are the key roles and responsibilities?

Sample Text

The software program director and supporting managers perform the following tasks:

- Approve all COTS and other reusable software components.
- Approve the product line common components plan or strategy.
- Approve all COTS and other reusable software components upgrade plans or other appropriate plans.
- Coordinate the execution of COTS and other reusable software component decisions through the Software Configuration Control Board (SCCB).

The software program director is supported in COTS and other reusable software component management by the software management organization, the lead architects, engineers, and the software teams.

6.1.4 Contracts and Procurement

Describe the key roles and responsibilities of the contracts and procurement organization in managing COTS and other reusable software components.

- What are the key roles and responsibilities?

Sample Text

The contracts and procurement organization performs the following tasks:

- Negotiate the full range of reusable software products and services required.
- Record and track license renewal dates in order to begin renewal negotiations in timely fashion.
- Notify program management of any change of status or developing problems with a reusable software component provider.
- Verify that contract stipulations related to reusable software components are met (e.g., latest version of source code in escrow, charges are consistent with contract, negotiated support provided).
- Assist in meeting all commitments made to the component provider.

6.2 ENGINEERING

Describe the roles and responsibilities in managing COTS and other reusable software components from an engineering perspective.

- What are the roles and the responsibilities of the engineering team and related teams?

Sample Text

The engineering team includes

- engineering leadership
- software engineering personnel
- reuse component managers
- component suppliers and vendors

Roles assigned to these team members are identified in the following sections.

6.2.1 Engineering Leadership Team

Describe the engineering leadership team's roles and responsibilities in managing COTS and other reusable software components.

- What are the roles and the responsibilities of the engineering leadership team?

Sample Text

The engineering leadership team (chief engineer, chief architect, chief software architect, and chief software engineer) is responsible for performing the following tasks:

- Ensure product line assessments are performed.
- Identify common capabilities to be developed and used across systems.
- Participate in planning for the development of common capabilities.
- Ensure that evaluations based on the Architecture Tradeoff Analysis Method® (ATAM®) and other architecture evaluations address reusable software components.
- Identify requirements and architecture changes that enable the use of COTS and other reusable software components.
- After reviewing evaluation criteria, provide recommendations to program management.

6.2.2 Software Team Engineering Personnel

Describe the software team's roles and responsibilities in managing COTS and other reusable software components.

- What are the roles and responsibilities of the software team's engineers and architects?

® Architecture Tradeoff Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Sample Text

The software teams' engineers and architects are responsible for the following tasks:

- Levy common COTS and other reusable software components requirements on their suppliers.
- Assign approval authority for COTS and other reusable software components to an appropriate designee.
- Ensure common COTS and other reusable software components are implemented properly in supplier products.
- Assist suppliers in reusable software evaluations.
- Support the requirements and architecture development activities as well as other architecture evaluations for making software reuse decisions.
- Support product line analysis activities.
- After reviewing evaluation criteria, provide recommendations to the software program director.

6.2.3 Reuse Component Managers

Describe the reuse component manager's role and responsibilities in managing COTS and other reusable software components.

- What are the responsibilities of the reusable software component manager?

Sample Text

Reusable software component managers represent a central point of focus for activities involving reusable software components.

The responsibilities of reusable software component managers include the following:

- Act as a central point of contact for dissemination and maintenance of information about a reusable software component.

- Assist engineering and other personnel in understanding and following expectations regarding processes related to a reusable software component.
- Work with management personnel to plan for and manage a reusable software component.
- Oversee the timely creation and review of all data and reports related to a reusable software component.
- Organize required meetings and reviews related to a reusable software component.

6.3 SUPPLIERS AND COTS VENDORS

Describe the suppliers and COTS vendors and their roles and responsibilities in managing COTS and other reusable software components.

- Is there a difference between a supplier and vendor for your program? Do you have different types of suppliers?
- What are their responsibilities?

Sample Text

A Component Supplier (Supplier) is responsible for providing a capability as directed by a Software Team or Program Management. The Supplier is tasked with providing the capability in the most appropriate manner based on the goals of the responsible Software Team and Program Management. Thus, a Supplier may provide the capability by reuse or by implementation of new software as appropriate.

Suppliers are normally chosen based on specific experience with the type of software to be provided. They will sometimes suggest that specific components they developed for other uses by modified or customized to achieve the Software Team/Program Management goals. These modified components are not normally considered “COTS” because they are not widely marketed and sold to multiple customers.

COTS Component Vendors, on the other hand, respond to market forces and are less likely to alter their products to meet specific requirements of individual customers. In fact, even when COTS vendors offer to make specific changes for such customers, such changes often move the resulting versions away from the common COTS baseline and can complicate maintenance.

6.3.1.1 Component Suppliers

Describe the key suppliers and their roles and responsibilities in managing COTS and other reusable software components.

- Who are the key suppliers? Where is this role defined?
- What are the key roles and the responsibilities for the roles?
- How are they different or the same from other programs in your organization?

Sample Text

Suppliers are responsible for performing the following tasks regarding component reuse:

- Evaluate potentially reusable software components using the techniques and processes defined in this document and in the SDP.
- Participate in defining requirements for common components to support the product line development approach.
- Satisfy the common requirements to enable a component's use across a program if a supplier is selected to implement the common component.
- Submit Component Evaluation Records (CERs) and enter data, including source code, concerning components into the software item (SI) database. (See Appendix E.)
- Cooperatively use common components when identified.
- Upgrade COTS software components to the common version per the schedule.
- Identify and report COTS and other reusable software components' data to facilitate approval and upgrades.

6.3.1.2 Component Vendors

Describe the key vendors and their roles and responsibilities in managing COTS and other reusable software components.

- Who are the key vendors? Where is this role defined?
- What are the key roles and the responsibilities for the roles?
- How are they different or the same from other programs in your organization?

Sample Text

COTS software vendors produce a product for sale in the marketplace that conforms to the characteristics of a COTS software product defined in Section 3. Further, vendors release products and upgrades that support their marketing strategy. This means the following:

- Vendors decide how a product works. If the program is not comfortable with the way a product functions, there are several options: adjust program expectations to align with the product, choose another product, or custom code a component.
- Vendors decide when patches, upgrades, and releases are made available. These schedules are rarely in alignment with the needs of the program. Refer to Section 9.6, Upgrade Management, for more information.
- Vendors use different licensing schemes. It is the responsibility of the program to understand the implications of the licensing scheme on program architecture and cost models, and maintenance. Refer to Section 9.4, License Management, for more information.

Component Vendors are required to adhere to the stipulations of contracts established between them and Component Suppliers, Software Teams, and the Program. Typically, these stipulations cover areas such as:

- licensing
- costs
- delivery schedules
- upgrades
- notification
- documentation
- escrow
- training

For each reuse component, a Reuse Component Manager should be designated. The roles and responsibilities of the Reuse Component Manager are listed in Section 6.2.3, Reuse Component Managers.

7 Process Artifacts

Describe the artifacts that are produced in accordance with the CRSMP.

- What information is captured in the artifacts?
- How are they related to other program artifacts?

Sample Text

In addition to identifying processes, the CRSMP defines six critical artifacts that document the current state of knowledge about a component and its use. These artifacts include

1. Make-Buy Decision Report
2. Component Evaluation Record
3. Reuse Evaluation Analysis Report
4. Component Life-cycle Plan
5. Component Health Check Report
6. Software Reuse Item Database (SRIDB)

The following sections define the high-level information contained in these artifacts. Detailed information for several of the artifacts can be found in the appendices as noted.

7.1 MAKE-BUY DECISION REPORT

The Make-Buy Decision Report document describes the strategy selected for fulfilling requirements for a portion of the system by either developing or procuring a reusable software component.

- What is the purpose of the report?
- What information is contained in the report?

Sample Text

The Make-Buy Decision Report documents the strategy for meeting a set of requirements—either create the component, or "buy" it, meaning "to procure it in some manner other than creating it."

The authors of the Make-Buy Decision Report perform the following tasks:

- Describe the component's niche.
- Identify who performed the analysis and the stakeholders who were consulted.
- Describe the major criteria that influenced the results and identify how these criteria were selected.
- Identify the results of a market survey, including the components available and market stability, appropriateness, and direction.
- Analyze the various make-buy alternatives and recommend a strategy.

7.2 COMPONENT EVALUATION RECORD

Describe the Component Evaluation Record (CER) document in general terms.

- What is the CER?
- What information does it contain?
- How is it related to the overall process?

Sample Text

The CER documents the evaluation of a component. CERs can be created during component screening or at other points during consideration of a component. CERs created during component screening are updated with additional information to reflect more detailed criteria employed during subsequent process steps, such as the component evaluation process. For components with no previous screening activity or for components that are being considered for an alternate purpose, a new CER is created. See Appendix E for more information.

The purpose of a CER is to

- Identify participants in the evaluation process.
- Identify the component being considered, including versions, configuration settings, and tailoring.
- Identify the niche that the component is being evaluated to fill.
- Identify the criteria considered during the evaluation.
- Document the performance of the component against those criteria.
- Document additional information about the component and component provider uncovered during evaluation.
- Analyze the impact of component use on the system (e.g., system capabilities, architecture, and maintainability).

7.3 REUSE EVALUATION ANALYSIS REPORT

Describe the Reuse Evaluation Analysis Report (REAR) document.

- What is the REAR?
- What information does it contain?
- How is it related to the overall process?

Sample Text

While an individual CER documents the evaluation of a single component, the REAR compares several components and their impact on the system. A REAR is generated following several individual evaluations and enables/aids consideration of the alternatives provided by those components for cost, schedule, architecture, design and other factors. See Appendix F for more information.

The REAR includes the following information[1]:

- summary of the evaluation processes and components considered
- comparison of alternatives and the selection made
- analysis of the impact on the selection on requirements and requirement weighting
- for the selected component (may reference information in the CER as appropriate)
  - description of component capabilities, assumptions, and limitations
  - description of needed customization, modification, porting, wrapping, etc.
  - cost and schedule estimates and the basis for them
- prioritized risks and risk-reduction activities
- implications of the selection for
  - quality attributes (such as performance, security, safety)
  - project and risk management and planning
  - engineering activities including architecture, design, integration, and test
  - system infrastructure
  - deployment, license management, training
  - training
- deficiencies in assessment methods and overall confidence

7.4 COMPONENT LIFE-CYCLE PLAN

Describe the Component Life-Cycle Plan (CLP) document.

- What is the CLP?
- What information does it contain?
- How is it related to the overall process?

[1] Note that information included in the CER or other reports can be referenced rather than duplicated.

Sample Text

The CLP documents the strategy for maintaining a component in good working order. The CLP includes the following information:

- life-cycle responsibilities
- existing or anticipated milestones, increments, phases, cycles of the component or of the system as it affects the component
- budgets and schedules
- approaches to managing releases and emergency updates
- data to be gathered about the state of the reusable software component within the system (e.g., changes to viability, developing problems and mismatches)
- configuration management of components and component versions, patches, adaptation code (e.g., tailoring, parameters, and schemas), baseline data
- training and support
- relationships with component providers
- license management
- component verification for releases and patches
- problem reporting and tracking
- contingency strategies

7.5 COMPONENT HEALTH CHECK REPORT

Describe the Component Health Check Report (CHCR) document.

- What is the CHCR?
- What information does it contain?
- How is it related to the overall process?

Sample Text

The CHCR provides a periodic analysis of the status of a component. The CHCR contains the following information:

- status of the component provider, including financial health, ability to meet expectations and schedules, and plans that can affect component use within the system
- status of relationships with the component provider, including recent communication and developing problems
- summary and analysis of metric data collected for the reusable software component
- summary of findings of market-watch activity as it relates to changes in the direction of the market, positioning of the component, and status of the vendor
- information from software teams and users regarding evolving expectations for the component

7.6 SOFTWARE REUSE ITEM DATABASE

Describe in general terms, the role and contents of the Software Reuse Item Database (SRIDB).

- What is the role of the SRIDB?
- What type of information is included in the SRIDB?
- Is the information dynamically or periodically updated over the life of the program?

Sample Text

The program uses the SRIDB to record pertinent information about reusable software components. It is a repository containing such information generated and saved in various reports. In addition, the SRIDB maintains relationships with various databases that contain additional information about components, including risk management, configuration management, problem tracking, and measurement databases.

Figure 2 contains a diagram depicting the SRIDB and the different databases to which it is connected.

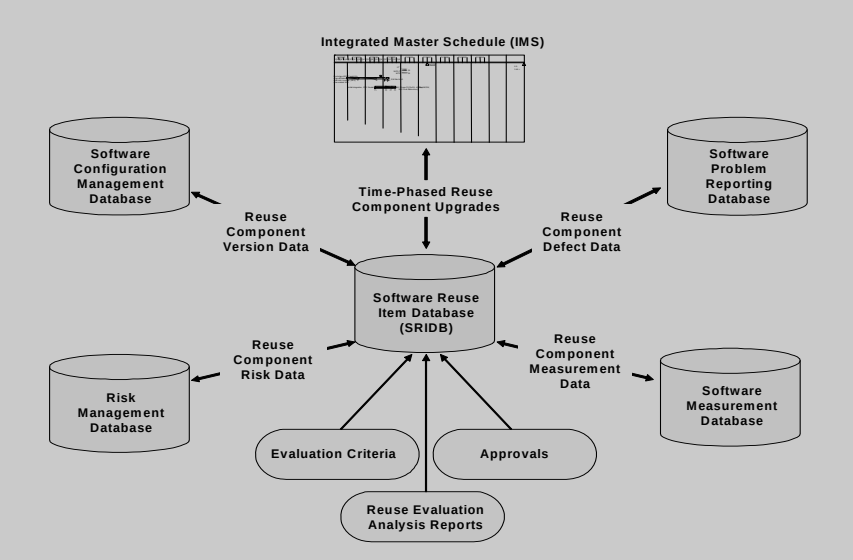

Figure 2: Managing Reusable Software Component Data

7.7 REUSE COMPONENT MATRIX

Describe the contents of the Reuse Component Matrix (RCM).

- What is the role of the RCM?
- What type of information is included in the RCM?
- Is the information dynamically or periodically updated over the life of the program?

Sample Text

The RCM identifies high-level information required to manage reusable software components in the program.

Information contained in the RCM includes the following items:

- identity of a component and all documents related to that component (can be links to other databases)
- identity of the individual who has primary responsibility for overseeing activities related to that component (the reuse component manager)
- identity of all suppliers and modules that use the component
- identity of all suppliers and modules affected by use of the component (this is often a larger set that is represented by the set of users due to indirect effects)

8 Process Descriptions

Note: The process descriptions contained in this section are based on a program adopting the WinWin Spiral development model. Your process descriptions will depend on the development model used by your program.

Provide sufficient detail about CRSMP processes to allow organizations to begin planning and staffing for execution.

- What are the goals of the processes?
- What are the details of specific processes?
- Who are the participants in the processes?

Sample Text

There are four life-cycle phases associated with the WinWin Spiral model: *inception*, *elaboration*, *construction*, and *transition*. Specific process guidance is provided for spiral stages and for life-cycle activities. However, the nature of the model implies that many of the activities in these phases may occur multiple times prior to a release.

In addition, software teams and their suppliers will likely execute different phases of the model at different times. Thus, one software team may have progressed to the point of transitioning a robust capability, while another software team may still be identifying baseline requirements. This staggered nature of process execution reflects the reality of developing large, complex systems. It is not easily diagrammed or documented.

However, the information provided in this document must follow linear page order. As an organizational principle, we elected to place specific processes within life-cycle phases. In addition, life-cycle processes (i.e., processes that are required throughout the use of a component) are placed within the transition phase, although they are ongoing processes that affect the work carried out in all phases.

The selected documentation order does not imply that a given process is only executed within a specific spiral phase, nor does it preclude the process from being executed during other phases of

the life cycle. The purpose is to place processes where they often occur within the following four spiral phases of Inception, Elaboration, Construction, and Transition, as indicated.

Inception Phase

- Identify common reusable software components.
- Categorize reusable software components.
- Conduct make or buy decision-making.
- Identify and manage reusable software component requirements.

Elaboration Phase

- Plan evaluation activities.
- Tailor evaluation criteria.
- Evaluate legacy (source code) software components for reuse.
- Evaluate COTS and similar software components for reuse.
- Evaluate software components that implement critical requirements.

Construction Phase

- Design
- Integrate
- Test

Transition Phase

- Release component

In addition, the following life-cycle activities occur during all life-cycle phases. Section 9, Managing the Life Cycles of COTS and Other Reusable Software Components, provides process guidance for these activities.

- life-cycle planning
- risk management
- impact analysis of requirements changes
- license management
- problem reporting and management
- upgrade management
- configuration management

- market watch
- component tracking and status review
- training and support
- deployment planning
- cost estimation and modeling
- reusable software component provider relationships
- reusable software component metrics
- health check

8.1 INCEPTION PHASE

Describe the goals of the inception phase and the processes executed during this phase.

- What are the goals of the inception phase?
- What processes are executed?
- Who are the participants in these processes?

Sample Text

The goal of the inception phase is to achieve concurrence among affected stakeholders on the life-cycle objectives for the solution. The inception phase establishes the feasibility of the solution through the business case that shows that one or more candidate solutions exist.

The following activities often occur during the inception phase:

- Classify components according to the extent of reuse and scope of the impact.
- Decide to make or buy systems, subsystems, and components.
- Determine detailed requirements for the component.
- Nominate, evaluate, and recommend candidate components.

8.1.1 Identifying Reusable Software Components

Describe how reusable software components are identified and how proliferation of similar components is avoided.

- By what processes are shared, reusable software components nominated for use?
- Who are the participants in those processes?

Sample Text

Strategic component reuse requires that suppliers, software teams responsible for modules of the system, and system integrators understand the capabilities that are common across their segments of the system. The architecture team leads a top-down effort to identify the common capabilities and components that will become elements of the software product line.

Analyzing the system for product lines and common components means analyzing the requirements, architecture, and desired capabilities. Software requirements and architecture development activities reveal patterns and commonalities that are examined for potential inclusion in software product lines. As this analysis becomes more refined, common capabilities and standard configurations of those capabilities mature. The common capabilities are used to identify common components.

The architecture team provides recommendations for implementing common components. Some common components may be mined from existing software or other reusable components. For common reused components, the architecture team identifies requirements and indicates what modifications they require to be part of the product line as appropriate.

In addition to top-down identification of common reusable software components, such components can be identified in a bottom-up manner by subcontractors and suppliers. As potential reusable software components are identified, they are screened to determine whether they should be shared across multiple segments of the system. Subcontractors and suppliers can nominate a component for elevation to “shared” status. The appropriate architectural authority will determine whether a component will be shared within a segment, across multiple segments, or across the entire system.

8.1.2 Categorizing Reusable Software Components

Identify categories of reusable software components relevant to the program.

- What categories of reusable software components exist?
- What is the basis for those categories?
- How are components classified according to those categories?
- How do the roles and responsibilities of major parties vary by category?

Sample Text

During architecting and engineering activities, suppliers and software teams analyze each COTS and other reusable software component to determine whether the impact of the component is limited to a supplier system or whether it has broader impact within a module or across modules.

Determining Impact of a Reusable Software Component

Concerns that influence whether a software component is isolated to a supplier system or whether it has broader impact include the following:

- exposed interfaces that are made available to other parts of a module or other modules
- data provided to or received by the component from other parts of the module or other modules
- influence on or expectations regarding memory, timing, and control sequencing of other parts of the module or other modules
- influence on or expectations regarding quality of service attributes such as security, safety, and reliability of other parts of the module or other modules
- unusual, unexpected, or unique constraints on the development, testing, or maintenance environments for the module or for the system
- influence on or expectations regarding the order or strategy of implementation, integration and test, or maintenance of other parts of the module or other modules
- influence on or expectations regarding configuration dependencies of other parts of the mod-

ule or other modules

- supplier risk evaluation (for example, size, or viability)
- unique licensing situations (as with, for example, open source software)

Categorizing Reusable Software Components

Reusable software components are categorized according to the following groups:

- Supplier Reuse–The component is used by a single supplier and has no effect outside of the systems produced by that supplier.
- Module Reuse–The component is used by multiple suppliers within a software team. It is not used by other software teams and has no effect on the systems produced by other software teams.
- System Reuse–The component is used by or has effect on multiple software teams.

The software team or supplier-recommended categorization of the extent and effect of the reusable software component is included in the SRIDB.

Review of reusable software component categorizations occurs at least one level above the scope of the component impacts, and recommendations for categorization are made to the engineering leadership team. For example, a higher authority (in this case called a software team) reviews components that have no expected effect outside of the capability provided by a supplier. The software team considers the supplier's suggestions and recommends a component categorization as appropriate. In all cases, the engineering leadership team exercises final approval authority.

Table 1 identifies the scope of impact relationships among the reuse component manager, reviewing authority, and final approval authority.

Table 1: Categorization of Reusable Software Components Based on Scope of Impact

Scope of Impact	Component Manager	Reviewing Authority	Final Approval Authority
Supplier reuse	Supplier designate	Depending on the scope, it could be the software team, software architect, or engineering leadership team	Engineering leadership team SCCB
Software team reuse	Software team designate	Engineering leadership team	Engineering leadership team SCCB
System reuse	Engineering leadership team designate	Engineering leadership team	Engineering leadership team SCCB

The reviewing and final approval authorities may solicit input from potentially affected stakeholders, including other suppliers and software teams, prior to taking action. Actions that can occur based on a component's categorization include the following:

- Approve as originally proposed.
- Re-categorize if the proposed category is either too narrow or too broad.
- Reject the component from further consideration.

Information about the how components are categorized is also maintained by the system architect as part of a reusable software component matrix.

8.1.2.1 Designating a Reusable Software Component Manager

Each component considered for reuse is assigned a reuse component manager. If it is a supplier reuse component, the supplier appoints a reuse component manager. If it is a software team reuse component, the software team architect appoints a reuse component manager. The reusable software component manager is normally chosen from the group with the most significant use of the component. If it is a system reuse component, the engineering leadership team collaborates with affected groups to appoint the reuse component manager, normally from the group with the most significant use of the component. In addition, the engineering leadership team forms a cross-group team as needed to oversee architectural, design, and implementation issues regarding the component.

The reuse component manager ensures that information about reuse components is available for sharing across a module and multiple module groups via the RCM, SRIDB, and other mechanisms. Intra- and inter-module group correspondence, analysis, discussion, and decision making regarding software team and system reuse components are logged and tracked.

8.1.2.2 Roles and Responsibilities for Categorization

Supplier Responsibilities

Suppliers are responsible for performing the following tasks regarding reusable software component categorization:

- Identify the preliminary extent of reuse and scope of impact for a component.
- Designate a reuse component manager for supplier reuse components.
- Designate a reuse component manager for software team reuse and other reuse components as directed by the engineering leadership team.

Software Team Responsibilities

Software teams are responsible for performing the following tasks:

- Designate a reuse component manager for software team reuse components.
- Analyze supplier designations of the extent of reuse and scope of impact of a component.
- Analyze the impact of components suggested by other modules as directed by the engineering leadership team.
- Execute the risk management process defined in Section 9.2, Risk Management.

Engineering Leadership Team Responsibilities

The engineering leadership team is responsible for performing the following tasks:

- Designate a reuse component manager for system reuse components.
- Review software team reuse and system reuse component designations, and approve all component designations.
- Create cross-module oversight teams as appropriate for system reuse components.
- Direct software teams to complete module impact analyses for components as appropriate.

Refer to Section 9.3, Requirements Management, for more information.

- Maintain the reusable software component matrix.
- Resolve conflicts by addressing the competing interests of software teams.

8.1.3 Make-or-Buy Decision Making

Define the process for determining whether a software component should be developed or reused. Only determine a strategy, not necessarily a specific component.

- What is the make or buy decision?
- What are the responsibilities of those involved?

Sample Text

The make-or-buy decision determines the strategy for meeting a specific set of requirements–either to build a custom component or reuse an existing component. Note that in many make-buy decisions, the actual component to be used will not be determined. Where a make-buy decision leads to selection, processes for component evaluation must be followed.

The make-or-buy decision-making process is primarily vested with software teams working with suppliers and should be accomplished as early as possible to minimize rework and unnecessary cost. Decisions are subject to review by the appropriate decision-making body.

Software Team Responsibilities

The software team's tasks regarding make-or-buy decisions include the following:

- Identify component to make or buy.
- Identify, document, evolve and manage requirements.
- Analyze and make recommendations with supporting rationale.
- Prepare a Make-Buy Decision Report.
- Collaborate with other affected stakeholders, for example, other software teams, regarding change management.
- Distribute the Make-Buy Decision Report to the appropriate engineering review board.
- Submit a make-or-buy review.

- Conduct an independent review for impact.
- Document component disposition and recommendations.

Software Configuration Control Board Responsibilities

SCCB responsibilities include the following tasks:

- Review and act upon recommendations.
- If differing opinions are voiced, determine the effect of alternatives and formulate recommendations.
- When applicable, update baseline configuration.
- When applicable, prepare necessary implementation measures in anticipation of customer concurrence and authority to proceed.
- Distribute make or by recommendations for review and approval.

8.1.4 Identifying and Managing Requirements for Reusable Software Components

Define the processes for identifying and managing requirements for reusable software components.

- What are the component requirements and how are they defined?
- How are requirements approved?
- How are changes to requirements managed?

Sample Text

Program success depends on consistently defining and evaluating module requirements and architecture with the overall system requirements and architecture in mind. Since the requirements for specific modules and the system are iteratively defined by using a spiral process, the requirements defined for reusable software components must be equally flexible. Maintaining flexibility requires negotiation of requirements, architectural alternatives, and reusable software components among software teams.

8.1.4.1 Defining Reusable Software Component Requirements

Sample Text

Defining the specific requirements for reusable software components occurs in the context of the overall spiral process used to define program system-level requirements. The early phases of the process focuses on understanding the functionality to be developed, thereby establishing a sound foundation for implementing the required functionality within the planned cost and schedule. Phases use prototypes and models of high-risk elements to understand functionality and to support feasibility analysis. Many of these prototypes and models may involve reusable software components.

Although the process for requirements definition is presented in a linear fashion, it is not intended to be a sequence of steps, but involves multiple proposals from software teams and the software architects with iterations to resolve questions and address risks that surface. The component requirements definition process includes the following activities:

- Software teams identify preliminary requirements for reusable software components. The requirements include functional and non-functional capabilities of reusable software components and desired characteristics of suppliers.
- Software teams document preliminary reusable software component requirements in accordance with requirements documentation processes as specified in the program's SDP. An entry for the component is created in the SRIDB. The component is classified according to potential effect on other program modules.

8.1.4.2 Vetting and Approving Reusable Software Component Requirements

Sample Text

The processes used to vet and approve reusable software component requirements differ, depending on component categorization as supplier reuse, software team reuse, or system reuse. Supplier reuse and software team reuse components are managed by the software team and associated SCCB. System-wide reuse components are managed by the program's SCCB.

The managing organizations develop strategies to ensure that all affected stakeholders are represented in deliberations regarding requirements for software team reuse and system-wide reuse components.

When conflicts or risks arise, or when the effect of proposed requirements on other modules is unclear, the system architect or appropriate software team architect executes the risk management process. Refer to Section 9.2, Risk Management, for more information.

8.1.4.3 Modifying Baseline Reusable Software Component Requirements

Sample Text

Changes to baseline requirements for "supplier reuse" or "software team reuse" components are managed according to the change management process of the responsible organization. However, changes also should be analyzed by the software team and by the system architect for impact on other modules.

- Where the changed requirement will affect other suppliers and/or other modules, the status of the component will be changed to software team reuse or system reuse as appropriate and all suppliers and software teams will analyze impact per Section 6.3 Requirements Management.
- Where the changed requirement leads to changing the component status to system reuse, management of the requirement will transfer to the engineering leadership team and program configuration control board.

For all changes to baseline requirements for system reuse components, the system architect will initiate a component requirements vetting and approval process per Section 9.3 Requirements Management.

In addition, suppliers and software teams analyze the potential effect of requirements changes on reusable software components by using the impact analysis process. Refer to Section 9.3, Requirements Management, for more information.

8.2 ELABORATION PHASE

Define the processes for selecting reusable software components.

- What planning is necessary for completing reusable software component evaluation activities?
- What specific strategies and processes will be employed?

Sample Text

The outline of the block delivery is detailed during the elaboration phase. Reuse components are selected and requirements are updated to reflect any new understanding about components and the system.

All candidate reusable software components are subject to detailed analysis before a specific component is selected for use within the program, as outlined in the SDP. Suppliers, software teams, and the engineering leadership teams select appropriate stakeholders, evaluation criteria, and evaluation strategies to reduce risk that a component or component provider fails to perform as expected after being selected.

For supplier reuse components, the supplier generally leads the evaluation effort. However, for software team reuse components where multiple suppliers are affected, the software team may choose to lead the evaluation, or may delegate responsibility to specific suppliers to lead the effort. Likewise, for system components, the engineering leadership team may lead the evaluation or designate a particular software team to lead an evaluation team that includes significant stakeholders from other software teams.

The following processes are conducted during the elaboration phase:

- Plan evaluation activities.
- Tailor evaluation criteria.
- Evaluate legacy reuse components.
- Evaluate reusable COTS components.

- Evaluate critical requirement components.
- Develop life-cycle plan.

These processes are described in the following sections.

8.2.1 Planning Evaluation Activities

Describe the planning required to evaluate COTS and other reusable software components.

- Who develops the Evaluation Plan (EP)?
- What information must be included in the EP?
- Who approves the EP?

Sample Text

The organization responsible for an evaluation (supplier, software team, or engineering leadership team) develops an evaluation plan (EP) that outlines the manner in which the evaluation is to be accomplished. Plans address the following topics:

- goals of the evaluation
- personnel performing the evaluation
- constraints on the evaluation (time and system constraints)
- relevant stakeholders
- criteria and mechanisms for performing evaluation
- prioritized/weighted criteria for performing evaluation
- strategies for consolidating results

The EP for supplier reuse components is approved by the appropriate software team. The evaluation plan for software team reuse and system reuse components is approved by the engineering leadership team. The approved EP normally becomes part of the CER.

8.2.2 Tailoring Evaluation Criteria

Identify the characteristics of evaluation criteria and how they will be generated.

- What are evaluation criteria?
- What sources are available for evaluation criteria?
- How is the relevance of example criteria established?

Sample Text

Evaluation criteria differ from component requirements in several ways: First, evaluation criteria address characteristics beyond functional capabilities defined at the system requirements level. For example, expectations regarding quality attributes may affect component evaluations. Second, evaluation criteria should identify the measurement or rating to be used (in other words, they should be capable of being measured).

Previous sets of evaluation criteria may provide useful information for subsequent evaluations. However, evaluation criteria created for one evaluation may not be appropriate in a subsequent situation or evaluation. Each time an evaluation is performed, the evaluation criteria must be re-considered.

Tailoring evaluation criteria addresses the variable contexts into which COTS and other reusable software components are placed within the system. No two situations are likely to be identical in terms of functional capabilities needed, quality characteristics required, and the other interacting system components to consider. Particular emphasis for a given evaluation is placed on the critical characteristics and component interactions required for the specific context.

Appendix E provides a Component Evaluation Taxonomy that identifies many potential categories and items for evaluation. However, the taxonomy is not intended to be and should not be used as a checklist. It is used to inform the reusable software component manager of many considerations that can be relevant to component evaluation.

When considering the aspects of reusable software components represented in the Component Evaluation Taxonomy, evaluators perform the following tasks:

- Identify the taxonomy categories and entries considered relevant to a specific evaluation and provide supporting rationale.
- Provide supporting rationale for taxonomy categories or entries considered not relevant to a specific evaluation.
- Define additional categories and criteria where appropriate.

When determining whether other categories or entries are relevant, consider the following factors:

- reusable software component category (e.g., supplier reuse, software team reuse, or system reuse)
- critical requirements and considerations (for example, safety and security)
- net-centric requirements
- component placement within the system
- manner in which the component is used
- expected nature of the interaction with the component vendor
- deployment and sustainment expectations including interoperability with legacy, NATO and other joint systems
- cost and complexity of the component
- other considerations identified by the architects, software teams, suppliers, or stakeholders

It is sometimes useful for evaluators to develop gate criteria as an initial screening device. These criteria can be used to determine whether a more extensive evaluation is warranted. Gate criteria are also valuable in the spiral process for making preliminary determinations about the availability of viable reusable software components. Common gate criteria include:

- measures of broad capability
- compatibility with other system components (for example, hardware, operating systems, previously selected components)

8.2.3 Evaluating Legacy Software Components for Reuse

Describe the strategy for evaluating legacy components.

- What approach will be used to evaluate legacy components?

Sample Text

When NDI products—particularly source code assets—require modifications for use in program systems, software teams apply the Options Analysis for Reengineering (OAR) approach for evaluating reusable assets. This approach helps determine whether candidate components are suitable for reuse, and what types of changes to the component or the interfaces are required by a software team or by other software teams. OAR provides an analysis of legacy components through six activities:

1. Establish mining context.
2. Inventory components.
3. Analyze candidate components.
4. Plan mining options.
5. Recommend mining option.
6. Evaluate mining options.

A team made up of suppliers and members of the software team and other relevant organizations perform the OAR. The method can be tailored, or an equivalent method can be used, if approved by the appropriate authorities.

8.2.3.1 Establishing Mining Content

Sample Text

The establish mining context (EMC) activity identifies the requirements for the specific components that are to be analyzed and provides pointers to the candidate components and their documentation. It also describes the programmatic and technical factors driving decisions. The candidate components are then analyzed and evaluated in later activities.

The team performs the following tasks as part of the EMC activity:

- Review goals, objectives, and drivers for the specific mining effort.
- Review requirements. Review relevant high-level requirements and how the requirements map from the system architecture to the software team architecture.
- Review and select component needs. Identify the component needs that will be addressed through this specific mining activity. These needs may be derived from the taxonomy included in Appendix D. However, only the most salient needs should be selected. The team may identify other needs. They can include characteristics such as functionality, interfaces, and code quality, technical constraints such as standards, language, or APIs, and quality attribute requirements.

8.2.3.2 Inventorying Components

Sample Text

The legacy components that can potentially be mined are screened in more detail. The team provides specific characteristics and criteria for evaluating the candidate components. Legacy components are evaluated based on these criteria and those that don't meet the criteria are disregarded. The results are in an inventory of candidate legacy components that meet requirements.

As part of this activity, the team performs the following tasks:

- Identify subsystems and software of potential interest and review the documentation for the selected software.
- Select high-value components for further consideration. During this step, the team scans subsystems and components, and compares them to the requirements identified during the EMC activity. Some potential components or subsystems may be omitted if they don't match the critical component requirements.
- Evaluate each remaining component on the full set of component characteristics and require-

ments.

- Document how each remaining component maps to the set of prescribed component characteristics.

8.2.3.3 Analyzing Candidate Components

Sample Text

The candidate set of legacy components are evaluated in more detail for their potential for use by the team. In particular, the types of changes that are required to each candidate component for inclusion in the module architecture (or the system architecture for components that are system reuse components) are identified. The team develops threshold criteria for the required characteristics.

The team performs the following tasks as part of this activity:

- screens each component in as much detail as necessary to determine if it meets the required threshold value; rejects those that do not satisfy these criteria
- identifies components that can be used “as is” or wrapped
- identifies components that need to be modified
- determines the types of changes that need to be made to each component, the cost and effort involved, the level of difficulty, risk, and the comparative cost and effort for developing each component from scratch.

For each component that passes the screening criteria, the team documents how the component scores relative to the threshold values. The team also documents the cost, risk and difficulty of performing the necessary rehabilitation for the component to conform to the software team’s architecture. If the component is categorized for system reuse, the team documents the variation points needed for it to conform to the system architecture.

8.2.3.4 Plan Mining Options

Sample Text

After the components are screened and the costs of making changes are estimated, alternative aggregations of components are considered that can provide greater or lesser value. The plan mining

activity enables the aggregation of components into different "options" and includes an analysis of the differential costs. In the case where only a small number of components are being considered, there may be only a single option. Alternative options for mining are developed based on schedule, cost, effort, risk, and resource considerations. A final screening of candidate components is conducted and the impacts of different aggregations of components are analyzed.

To plan mining options, the team performs the following tasks:

- selects promising components and develop criteria for performing a final screening based on project consideration. These criteria may include cost, effort, level of difficulty, and value provided to the target system.
- forms component options by developing criteria for aggregating components to form options. These criteria may include the number of components and impact on schedule, mapping between legacy system and target system needs, and value of the aggregated option provided to the target system.
- forms component aggregations as appropriate.
- determines comparative cost and effort. For each option, determine the comparative cost of developing the components included from scratch.
- determines level of difficulty and risk for each option. The team makes estimates of confidence at the 10-percent, 50-percent, and 90-percent levels as one indicator of risk. The rationales for the low and high reuse estimates are provided.

8.2.3.5 Recommend Mining Option

Sample Text

The team recommends the mining option or combination of options that can best satisfy team goals by balancing programmatic and technical considerations. Each mining option is evaluated. The team recommends an optimal option or combination of options. A summary report and justification for the recommended option is prepared.

To determine the recommended mining option, the team performs the following tasks:

- sets priorities of the drivers for selecting the options
- evaluates each potential option according to the priorities that were selected; identifies component needs
- identifies risks and confidence intervals in reusing the software or making changes
- presents findings and provide details of the options selected and their rationale

8.2.3.6 Evaluate Mining Options

Sample Text

The team evaluates the recommendations of each of the suppliers. An approach is chosen for a subsystem based on the criteria that was identified during the establish mining context activity. A summary report and justification for the selected option is prepared.

To evaluate mining options, the team performs the following tasks:

- reviews reports
- establishes criteria for reviewing the mining team reports. Criteria may include quality of analyses, completeness of results, consistency of results, and compliance with the reuse evaluation method.
- evaluates supplier proposals based on software team criteria that are satisfied, impact on software team cost and schedule risk, and supplier compatibility and capability
- evaluates the credibility of the supplier recommendation based on the evaluation criteria
- selects options from supplier reports; selects the combination of potential options that best meet the needs of the software team
- identifies component needs satisfied and complete the final list of component needs satisfied and not satisfied
- produces a summary report detailing the option(s) chosen and rationale for their selection

8.2.3.7 Collecting Metrics During Legacy Component Analysis

Sample Text

When analyzing the reuse potential of legacy components for which source code is available, the following metrics are collected:

- thousands of lines of code (KSLOC) of the source code
- equivalent KSLOC (EKSLOC), which is the cost to produce the modified source code
- benefit KSLOC (BKSLOC), which is the cost avoided as a result of reuse

8.2.4 Evaluating COTS Software Components for Reuse

Describe the processes used to determine whether COTS software components (and other components for which source code is not available) can be reused.

- What approach will be used to evaluate COTS and other components for which source code is not available?
- What are the steps to be followed?

Sample Text

The use of COTS and other non-source software components represents an opportunity to provide a rapid and cost-effective solution. However, if these components are selected based on insufficient or erroneous information, they can present a significant risk to the system.

Insufficient or erroneous evaluation can occur in several ways:

- The level of effort expended for evaluation is not commensurate with the importance of the component.
- New releases of the component and changes to system requirements do not lead to subsequent reviews of component viability.
- Insufficient consideration is given to the environment in which the component must execute and be sustained.
- Involvement of experts who understand the fielded environment is limited.
- The evaluation does not involve people having hands-on or other pragmatic experience with the product.

To encourage careful evaluation and selection, the process for evaluating COTS software components is derived from the planning, establishing, collecting, and analyzing (PECA) process. The PECA process can also be applied for evaluating other sorts of components for which source code is not available. Here, PECA is considered from the perspective of evaluating COTS software components.

The steps of the PECA process include

- planning the evaluation
- establishing criteria
- collecting data
- analyzing data

8.2.4.1 Planning

Sample Text

Planning tasks required prior to a COTS software evaluation were previously defined in Section 8.2.1, Planning Evaluation Activities. When planning the activity, the team develops clear goals for the evaluation and identifies the steps to be taken to determine whether a COTS software product meets program needs. The team also identifies relevant stakeholders, including end users or other field experts, system administrators or those charged with sustainment of the system, and system and software architects and designers. The team identifies previous decisions that impact the evaluation, selection, and use of the component.

8.2.4.2 Establishing Criteria

Sample Text

Criteria for evaluating a COTS software component are selected based on several guiding principals, such as

- significance to the particular component and system capability to be provided
- ability to discriminate between alternative COTS software capabilities in a cost-effective manner
- lack of component overlap to avoid possible duplicate costs and over-weighting of results
- accessibility in a straightforward, consistent, impartial, and unambiguous manner

A criterion consists of a clear statement of expectation for the COTS component, along with a means of assessing and assigning a value representing the component's compliance to requirements. Appendix D provides information about selecting potential COTS software component evaluation criteria. This appendix is included to provide general guidelines and encourage consideration of a range of criteria.

Where possible, the criteria are categorized as *non-negotiable* and *negotiable*. Non-negotiable criteria reflect those aspects or characteristics of a component or its provider that are essential for execution or sustainment of the program. If a COTS software candidate fails to meet expectations for a non-negotiable criterion, it is effectively disqualified from consideration.

Negotiable criteria reflect those aspects or characteristics of a component or its provider that have some degree of flexibility in terms of capability or other considerations. Negotiable criteria provide flexibility in the evaluation and selection process to consider alternate strategies for reaching the overall program goals.

In order to counteract the tendency of evaluators to consider most criteria non-negotiable and to maximize software reuse within the program, evaluators should provide rationale for the selected categorization of criteria.

When criteria weighting techniques are used, techniques applied should be defensible and account for the collected expertise of the range of stakeholders and evaluators. Techniques that eliminate inconsistencies in weightings, for example, pair-wise comparisons of attributes as reflected in Thomas Saaty's *Analytical Hierarchy Process* are preferred over less formal weighting techniques [Saaty 1980].

Criteria are documented as part of the CER or as part of the EP and provide

- clear statements of the capabilities expected
- metrics
- identification as non-negotiable or negotiable
- rationale for classification
- assigned weighting

8.2.4.3 Collecting Data

Sample Text

In the collection of data, efforts will be made to eliminate bias in the evaluation process by providing guidelines for all evaluators to follow as appropriate. Guidelines may consist of strategies for gathering data, and explanations of various rating categories or classifications, for example, "poor," "fair," and "good." This is particularly important for subjective ratings that are often part

of a reuse component evaluation.

Hands-on evaluation of potential reuse components within testbeds and prototypes is a preferred strategy. Such hands-on evaluations focus on the interactions between the candidate components, other candidate components as appropriate, and already-selected and constructed program components.

Data gathered during the evaluation process is captured and stored as part of CER.

8.2.4.4 Analysis

Sample Text

Analysis of data gathered during evaluation of COTS software components takes into account the following

- performance against evaluation criteria
- identification of criteria that are likely to change as program evolves
- analysis of the sensitivity of the evaluation to changes in criteria, also called sensitivity analysis
- gaps in capabilities provided by the candidate component
- approaches to bridging or otherwise mitigating gaps in capabilities
- relative rating of candidate components and associated bridging/mitigating strategies
- insight into system and component requirements
- insight into program architecture, design, and component selection and construction
- confidence in the evaluation process
- limitations of the evaluation process
- feedback and suggestions about the evaluation process

If a software component must satisfy safety-critical, mission-critical, security, or other critical requirements, appropriate analyses are conducted as described in Section 8.2.5, Evaluating Critical-Requirement Software Components for Reuse.

The evaluation of COTS software components is documented according to the CER template defined in Appendix E. Existing CER documents created during component screening are updated with new, more detailed information, and records of component evaluation criteria are extended

to incorporate detailed criteria employed during the component evaluation process. For components not previously screened, or for components that are being considered for an alternate purpose, a new CER is created.

8.2.5 Evaluating Critical-Requirement Software Components for Reuse

Define strategies used to evaluate critical-requirement software components for reuse.

- What techniques apply to evaluating critical components?

Sample Text

If a software component must satisfy safety-critical, mission-critical, security, or other critical requirements, appropriate analyses as described in the section titled “Strategies for Handling Critical Requirements” in the SDP are conducted.

However, some characteristics of reusable software components, such as the inability to access source code, can severely limit the mechanisms by which critical requirements can be evaluated. In these cases, the evaluating organization can submit an alternate plan for verifying the component. The plan must describe

- why the SDP cannot be implemented as specified
- the alternate strategy for verifying critical requirements, when, for example, source code is not available
- limitations placed on use of the component

The alternative plan is approved by the appropriate technical leads.

8.2.6 Evaluating Life-Cycle Impact

Define strategies for evaluating the life-cycle impact of reusable software components.

- What strategies are appropriate for evaluating life-cycle impact?

Sample Text

Typical software component evaluation strategies include consideration of life-cycle cost. While this data is important, it is not the sole life-cycle consideration reflected in component evaluation activities.

Other aspects of life-cycle implications of reusable software components that are considered include

- licensing, configuration, and release management
- component verification for releases and patches
- risk and problem reporting and tracking
- training and support

Additional considerations for the life-cycle support of reusable software components are identified in Section 7.4 Component Life-Cycle Plan and Appendix D.

8.3 CONSTRUCTION PHASE

Define the processes necessary to construct a version of the system using reusable software components.

Sample Text

The construction phase involves the design, implementation, integration, and testing of a release. Organizations involved in building parts of the system use standards-based APIs to facilitate the integration of COTS and other reusable software components. When standards-based APIs are not

available for facilitating integration, a number of adaptation approaches including modification, tailoring, and wrapping are available.

8.3.1 Design

Describe the strategy for incorporating reusable software components into the system.

- How are reusable software components documented?
- What provisions are made to isolate the reusable software component from other system components?
- How are design decisions regarding the reusable software component documented?

Sample Text

Whenever possible, suppliers document all COTS and other reusable software components according to the expectations for software design specified in SDP. Minimally, this documentation details component interfaces using appropriate modeling languages (e.g., UML, XMI, XML). In addition, for each COTS and other reusable software component, suppliers identify the following:

- critical quality of service expectations expected by and met by the component
- characteristics of allowable call sequences to the component and call sequences made by the component to other components

8.3.1.1 Isolation of Capability

Sample Text

A primary design goal when using COTS and other reusable software components is to isolate the rest of the system of systems from changes made to the component. This is particularly important for reusable software components that affect the execution of other components outside of a supplier's control or in other modules (e.g., software team reuse and system reuse components).

COTS and other reusable software components are isolated according the following guidelines:

- Standards-based APIs are employed whenever possible. Extensions to standard interfaces (when provided by components) are prohibited unless a waiver is granted by the engineering leadership team.
- When standards-based interfaces are not available, features of selected reusable software components are limited to those commonly available in other components from the same marketplace segment. The use of hidden, unique, or special features is prohibited unless a waiver is granted by the engineering leadership team (for system reuse components) or software teams (for software team reuse and supplier reuse components).
- Use of nonstandard interfaces or deviation from common practices involving standard interfaces is documented in the SRIDB.

Component interfaces employed within modules (software team reuse components) are subject to approval by the software team. Component interfaces employed for system reuse components are subject to approval by the engineering leadership team.

Entries for reusable software components that cannot be effectively isolated from the rest of the program system are entered in the risk management system as appropriate. This also applies for reusable software components for which full functional capabilities or qualities of service are unclear. Refer to Section 9.2, Risk Management, for more information.

8.3.2 Implementation

Describe the guidelines for preparing reusable software components for incorporation into the system.

- What sorts of component preparation are allowed?
- How are various pieces of code developed to adapt the component to the system and vice versa? How are the new sections of code treated?

Sample Text

The implementation step involving COTS and other reusable software components does not involve any direct modification to the core products. However, products normally require that parameters be set and sometimes that extensive customization be performed using built-in tools (e.g., pre- and post-processing of data, interface development, enhancement of basic capabilities).

Implementation for some other NDI reusable software components can involve a range of implementation activities from "as-is" use to modification of the component. Before the decision to modify any component is acted upon, give full consideration to the costs, benefits, and risks.

Regardless of the purpose or extent of new code developed to integrate COTS and other reusable software components, this new code is subject to requirements as specified for new code in the SDP. Additionally, the components and new code are subject to the requirements for modification, tailoring, and wrapping as specified in the SDP. If these requirements cannot be met, the integration process should be documented in the suppliers' or software team's SDPs.

8.3.2.1 Tailoring, Customization, Extensions, and Wrapping

Sample Text

For our purposes, no distinction is made between the terms *tailoring*, *customization*, and *extensions*. Each term refers to changes to "out-of-the-box" products to adapt them to fill a particular system need as specified in the SDP. Each change to out-of-the-box functionality can potentially add to the complexity of long-term sustainment of the component. Overuse of such changes has been disastrous for several programs.

Our goal is to minimize tailoring, customization, and extensions applied to COTS and other reusable software components. This is not always easy to accomplish because the use of COTS and other reusable software components creates conflicts between expectations regarding system capabilities and the capabilities and interfaces of reusable software components. Balancing expectations with the consequences of long-term sustainment is a challenge.

Additionally, COTS and other reusable software components may require *wrapping*, which is a technique used to provide a layer of isolation between the component and the system's environment. Wrapping in itself does not modify out-of-the-box component capability. Carefully considered wrapping can extend the capability of components and ease future upgrades. However, wrappers and their relation to components also have consequences for sustainment as components and system expectations change over time.

In order to meet the program goal of minimal tailoring, customization, extension, and wrapping suppliers perform the following tasks:

- Adhere to guidelines provided in the SDP regarding strategic reuse of existing software.
- Collect and report data quarterly regarding the volume, cost, and complexity of tailoring, customization, extension, and wrapping.
- Analyze the impact of tailoring, customization, extensions, and wrapping on life-cycle maintenance.

Software teams (for supplier reuse and software team reuse components) and the engineering leadership team (for system reuse components) perform the following tasks:

- Adhere to guidance provided in the SDP regarding strategic reuse of existing software.
- Review and approve all tailoring, customization, extension, and wrapping requests.
- Monitor data and work with suppliers to develop strategies to minimize tailoring, customization, extensions, and wrapping when necessary.

8.3.3 Integration

Describe expectations regarding integration of the reusable software component.

- When does integration occur?
- How is the integration strategy documented?

Sample Text

Integration of COTS and other reusable software components is not a one-time activity for the program. The spiral life cycle identifies multiple situations where new versions or potentially different components will need to be integrated. This continuous re-integration activity is driven by many factors, including better understanding of requirements, increasing or changed commitment to specific components, and new version releases. Each new integration may require adapting the system and associated data.

The SCCB communicates the timing of incorporation of the component into a build or multiple builds. Where standards-based APIs are not directly available for integration, a number of adaptation approaches including modification, tailoring, and wrapping are available. Expectations regarding the integration of software subsystems and system, including the software integration plan, are identified in the SDP.

For each COTS and other reusable software component, the following data regarding integration is maintained in the SRIDB:

- integration strategy
- rationale for the selection of the strategy
- steps involved in the integration activity
- impact on overall system capability and performance, system and subsystem architecture, design, integration, testing, sustainment, maintenance, system infrastructure, deployment, and configuration management
- dependencies on non-standard interfaces and data formats

The information in this report is used to develop integration plans. After the COTS or other reusable software components are integrated, they are tested along with the other components of the integration package. During software item qualification testing, COTS and other reusable software components' capabilities and performance are tested in coordination with other system capabilities and performance.

In addition, suppliers update the appropriate documents and databases as necessary:

- Problems found are recorded in the software problem reporting database.
- Identified risks are entered into the risk database (see Section 9.2, Risk Management).
- General lessons learned about the COTS and other reusable software component integration processes are entered into the software transition plans.

Additionally, information regarding integrating components is included the software version description (SVD) found in the SDP.

8.3.4 Testing

Describe the approach to testing reusable software components.

- What specific testing strategies will be employed?

Sample Text

Suppliers and software teams maintain sufficient testbed capabilities to verify reusable software components' capabilities, test new releases, and evaluate candidate or replacement products, providing a continuous development and integration process. Qualification testing is performed as specified in the SDP.

Suppliers verify adherence to critical quality requirements of each new release of COTS and other reusable software components included in a program release as described in the SDP. In addition, suppliers benchmark all new versions to assure that they perform within appropriate limits.

If testing indicates significant change in critical qualities or performance of a component, suppliers enter a risk in the risk database and notify software teams and the engineering leadership team as appropriate. Refer to Section 9.2, Risk Management for more information.

When system reuse components are employed, suppliers work with software teams, the engineering leadership team, and other appropriate labs to define appropriate strategies for testing component interoperability.

8.4 TRANSITION PHASE

Describe the approach to releasing a reusable software component to other software teams and users.

- What activities occur during the transition phase?
- What information is maintained about release of a reusable software component?

Sample Text

The transition phase encompasses activities to prepare a reusable software component for release to other suppliers, software teams, or as part of a program block release.

During transition, reusable software components may be refreshed with the current version of the product. Other reusable software components may be modified to eliminate defects and to satisfy evolving requirements. New COTS and other reusable software components may become available to satisfy software requirements. These and other activities related to the transition of COTS and other reusable software component transition activities are addressed in the software transition plan included in the SDP.

8.4.1 Component Release

Describe specific expectations regarding release of a reusable software component.

- What does the receiving organization need to receive in addition to the component?

Sample Text

Each software release containing a change to a reusable software component is considered a new version from the supplier and is prepared and documented according to the SVD as defined in the SDP.

The SVD also includes the following information for each COTS or other reusable software component:

- identification of the SRIDB entry related to the component
- identification of the component manager
- information about licensing
- information about changes to dependencies (e.g., versions, configurations) on hardware and other software
- description of required settings and tailoring

- description of changes to interfaces, processing, and mission-critical performance (see the SDP)
- changes to documented uses of the component and limitations and conditions placed on use
- copies of release notes from the software provider

For software team reuse and system reuse components, the following information is also included in the SVD:

- impact analysis identifying other suppliers that are affected by changes to reusable software components
- impact analysis identifying other modules that are affected by changes to reusable software components

8.4.2 Deployment Planning

Describe special concerns for deployment planning in this section.

- How do COTS and other reusable software components affect deployment planning?
- Are any special roles and responsibilities required?

Sample Text

In addition to deployment planning as specified in the transition to operations and transition to support plans, deployment strategies to address COTS and other reusable software components include guidelines for

- engineering and coordinating multiple suppliers' releases with the program's system releases
- tailoring reusable software components and generating data loads for site-specific requirements
- end-user support to ensure that users are knowledgeable in using the system (training) and have access to customer support that responds to their questions
- managing multiple fielded releases of reusable software components as well as the different

configurations of systems that contain these components

Reusable software component managers support program and technical management in formulating specific guidelines.

9 Managing the Life Cycles of COTS and Other Reusable Software Components

Describe the main COTS and other reusable software component life-cycle management activities not addressed previously in the CRSMP, including specific interactions with supporting program and engineering management approaches.

- What are the main COTS and other reusable software component life-cycle management activities that have not been addressed so far in this document?
- How does information contained in this document influence or affect other program and engineering management approaches?

Sample Text

COTS and other reusable software components create unique management challenges, since the evolution of the components and life-cycle milestones are outside the control of the program. Life-cycle activities are aimed at minimizing risk and maximizing the effective return from COTS and other reusable software components and are identified in this section.

9.1 LIFE-CYCLE PLANNING

Describe the life-cycle plan for reusable software components.

- When is the life-cycle plan developed?
- What is included in the plan?

Sample Text

Planning for the life-cycle impact of a reusable software component starts during component evaluation, when a range of potential affects from reusable software components are considered. The information gathered during evaluation becomes the basis for an initial life-cycle plan. Life-cycle planning continues during system elaboration, construction, and transition, when the life-

cycle plan for reused software components is improved and maintained.

Topics addressed in the life-cycle plan include

- existing or anticipated milestones, increments, phases, cycles of the component or of the system as it affects the component
- responsibilities for component planning, management, budget, and schedule
- license, configuration, and release management
- metrics
- training and support
- component verification for releases and patches
- problem reporting and tracking

Section 7.4, Component Life-Cycle Plan, contains additional information.

9.2 RISK MANAGEMENT

Describe the interactions between the information contained in the CRSMP and risk management activities.

- How do the management of COTS and other reusable software components affect risk management?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Risk management for COTS and other reusable software components occurs in the context of overall program risk management. The process of risk management is not significantly different, but the risks that arise from the reuse of software components often are unique. This is due to the lesser degree of control that the program has over COTS and other reusable software components controlled by outside influences. For reusable software components, the kinds of risks and the techniques used to mitigate them both differ. In addition, the timing of risks associated with COTS and other reusable software components may not be as predictable as those that occur dur-

ing the development of custom software code.

Typical activities performed to identify and address risks associated with reusable software components include

- creating and updating risk information in any tracking tools and updating risk management plans as appropriate
- initiating a spiral "loop," which often involves using a prototype or model to analyze alternatives, resolve risks, and establish commitments between parties.
- documenting conflict, risk reduction, and analysis activities, decisions, and commitments as appropriate
- updating appropriate supporting documentation such as requirements documentation

Each member of the team is responsible for identifying risks. The engineering leadership team directs the exploration of cross-module risks by using appropriate mechanisms such as benchmarking, modeling, and rapid prototyping.

9.3 REQUIREMENTS MANAGEMENT

Describe the interactions between the information contained in the CRSMP and the impact analyses of requirements changes.

- How does the management of COTS and other reusable software components affect requirement development and requirements management?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?
- How will you ensure that the impact analysis will addressed functional and non functional requirements as well as program constraints?

Sample Text

Section 8, Process Descriptions**,** discussed the identification and management of requirements associated with reusable software components during development of a system. However, requirements of complex, modern systems are rarely static and continue to evolve after the system

is deployed and until it is decommissioned.

Therefore, throughout the entire system life cycle, the appropriate staff members are responsible for analyzing the potential impact of requirements changes on COTS and other reusable software components within their purview. Likewise, personnel are responsible for analyzing the potential impact of changes to COTS and other reusable software components on the system and other program elements (e.g., effect on functionality, quality attributes, cost, and schedule). Each reusable software component manager is responsible for coordinating impact analyses and for disseminating information to all affected parties.

Impact analysis of requirements changes on COTS and other reusable software components takes into consideration

- effect of the change on existing capabilities and planned capabilities
- strategies for adapting to the requirements change where possible
- alternative solutions where adaptive strategies are not possible
- cost of implementing adaptive strategies and alternative solutions

Impact analyses of COTS and other reusable software components on the system and other program elements is critical to the success of program and takes into consideration the effect of the changes on

- existing and planned capabilities
- program context
- end-user processes

The impact analyses also identify alternative solutions and estimate the cost of implementing those alternative solutions.

When a change to a requirement creates a previously unidentified risk, the risk management process is executed.

9.4 LICENSE MANAGEMENT

Describe the strategy for managing licenses.

- What is your strategy for managing licenses?
- Are any special roles and responsibilities required?

Sample Text

The purpose of managing licenses is to control licensing costs, to ensure liability is assessed, to protect intellectual property, to obtain warranties, to specify remedies, and to ensure the consistent and continuous operation of the system.

The software management organization and the software teams, in conjunction with the component manager, examine the licensing option and cost information recorded in the SRIDB to determine the most effective options program-wide. For multiple-use components, when sufficient license quantities warrant, appropriate parties negotiate license discounts with the software vendor.

Licenses for other reusable software components are also managed to ensure adequate data rights and software support.

The SRIDB includes information about licensing restrictions and limitations that are managed by the responsible organization, in conjunction with the reusable software component manager. These restrictions and limitations are analyzed to ensure that they do not impact operation and support of program systems and capabilities.

9.5 PROBLEM REPORTING AND MANAGEMENT

Describe the interactions between the information contained in the CRSMP and problem reporting and management.

- How do the management of COTS and other reusable software components affect problem reporting and management?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Problems with reusable software components are reported and tracked by using software problem reports (SPRs), as specified in the SDP. In addition to factors covered by standard defect analysis, analysis of defects in COTS and other reusable software components takes into consideration

- potential and timing of vendor/NDI provider repairs
- likelihood of repairs being reflected in the COTS or other reusable software component baseline (e.g., all subsequent deliveries from the vendor)
- alternate strategies for repair that do not involve vendor cooperation (e.g., additional tailoring, wrapping, bypassing of defective functionality)
- potential for alternate components

Appropriate resolution of SPRs and other changes to reusable software components rely on all affected organizations to assess and report on the impact of the proposed changes on their systems.

For problems involving reuse components within a module (software team reuse), the problem is referred to the appropriate architect and module SCCB. For problems involving a system reuse component, the problem is referred to the system architect and appropriate system-level SCCB. The appropriate architect notifies all affected parties of the problem and solicits information on resolutions as appropriate. Resolution of conflict is directed by the appropriate architect and team.

Reusable software component managers are responsible for coordinating and consolidating impact statements for affected suppliers and software teams, as necessary, and for storing impact statements in the SRIDB.

9.6 UPGRADE MANAGEMENT

Describe the upgrade management process in general.

- What is your strategy for managing upgrades to COTS and other reusable software components?
- Are any special roles and responsibilities required?

Sample Text

The upgrade cycle for reusable software components is generally outside the control of the program. It must, however, be monitored and closely coordinated with testing and release blocking on the program. Changes to components are going to occur and the program must be ready to effectively manage changing components and systems.

9.6.1 Patches

Describe the upgrade management process with regard to software patches, specifically.

- What is your strategy for managing software patches?
- Are any special roles and responsibilities required?

Sample Text

When a patch for a reusable software component is released, the patch is evaluated under the direction of the reusable software component manager for the benefits it provides and the harm it inflicts. If the benefits outweigh the harm, the controlling software team authorizes the patch and communicates it through the SCCB.

9.6.2 Version Upgrade

Describe the upgrade management process with regard to software version upgrades, specifically.

- What is your strategy for managing software version upgrades?
- Are any special roles and responsibilities required?

Sample Text:

The reusable software component manager works with suppliers, software teams, and appropriate SCCBs to coordinate and consolidate version upgrade tasks. When multiple organizations or systems are affected by upgrades, the reusable software component manager consolidates the analyses of the parties.

New versions of COTS and other reusable software are analyzed by affected parties with regard to

- urgency of the change
- benefit, costs, and risks of the change
- past performance of the component following an upgrade
- impact of the change on the support and test environment
- proximity and relationship to expected end of life of the component
- impact on tailoring, customization, extensions, and wrapping
- impact on training
- impact on technical data and computer resources
- impact on functional capabilities and system qualities (e.g., performance, security, reliability, safety)

For critical requirements, the upgrade analysis includes evaluations required by the SDP.

In general, version upgrades should enhance the quality of the software, while maintaining existing functionality. However, the software team or engineering leadership team can make exceptions to this rule when (for example):

- the currently used version is no longer supported
- changes are forced due to interdependencies among components

When changes to a reusable software component are particularly complex and significant, the software team or engineering leadership team can consider alternate approaches to providing the service (e.g., alternate components, custom build).

9.6.3 Major Upgrades

Describe the upgrade management process with regard to significant software upgrades, specifically.

- What is your strategy for managing major upgrades?
- Are any special roles and responsibilities required?

Sample Text

Reusable software component managers identify major upgrades to reusable software components that are planned by component providers and notify affected parties. Information regarding major upgrades is presented during reusable software component status reviews. Analysis of the impact of major upgrades occurs as for version upgrades.

The reusable software component manager or other individual or organization identified by the supplier (for supplier reuse components), the software team (for software team reuse components), or the engineering leadership team (for system components) coordinates planning for component reevaluation, integration, and testing that occurs due to major upgrades.

9.6.4 Component End of Life

Describe upgrade management with regard to component end of life, specifically.

- What is your strategy for managing component end-of-life situations?
- Are any special roles and responsibilities required?

Sample Text

Special attention must be given when a reusable software component nears its end of life. The reusable software component manager is responsible for identifying end-of-life risks. The SPR mechanism is used to trigger impact analyses across all affected parties. The reusable software component manager provides a recommendation at the status review with regard to sustainment or replacement options.

9.6.5 Upgrade Schedule

Describe upgrade management with regard to upgrade scheduling.

- What is your strategy for managing the upgrade schedule?
- Are there any special roles and responsibilities required?

Sample Text

The software management organization analyzes the COTS software products, versions, and dependencies and then creates an upgrade plan. To create the upgrade plan, the team examines the impact on suppliers, schedule, and costs to optimize a solution across the program. The plan is then approved by the SCCB and implemented.

The software management organization also analyzes related upgrade schedules when developing the upgrade plan. The purpose is to synchronize the COTS and other reusable software component upgrade schedule with other upgrade activities.

9.7 CONFIGURATION MANAGEMENT

Describe the interactions between the information contained in the CRSMP and configuration management tasks.

- How does the management of COTS and other reusable software components affect configuration management?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Configuration management of COTS and other reusable software components occurs in the context of overall program configuration management. (Refer to the program's configuration management plan). However, rigor and sophistication of configuration management procedures and tools are required for

- impact analysis of new COTS software releases or new COTS software products
- development of one or more upcoming system builds
- integration and test
- system test
- multiple deployed baselines

Additional configuration management support is necessary for licensing information, multiple component versions, associated artifacts (patches, documentation, installation scripts, etc.), and market research.

The program also tracks specific components from the time they are first received through the full duration of their use in the system.

9.8 MARKET WATCH

Describe market-watch activities.

- What is your strategy for watching the market?
- Are any special roles and responsibilities required?

Sample Text

A market-watch activity is initiated for appropriate market segments as assigned by the software team architects. Reusable software component managers participate in market-watch activities for their assigned components.

The market-watch activity develops and maintains a market segment artifact that captures such characteristics as vendors, suppliers, and buyers participating in the market, COTS software components offered, processes automated, technologies represented, procurement strategies practiced, and competitive market forces. The market segment information artifact focuses on large-scale market dynamics rather than in-depth analysis of individual components.

The market segment artifact accumulates and organizes information used to perform the following tasks:

- Verify that the general marketplace represented by the reusable software component is appropriate for the project (e.g., appropriately stable, innovative, customer centered, organized, etc.).
- Verify that active buyers and users of the capabilities produced in the marketplace have needs similar to those of the project.
- Verify that active buyers and users of the capabilities produced in the marketplace are employing marketplace offerings in a manner similar to that anticipated by the project.
- Identify the strategies used by active buyers and users for acquiring reusable software components produced by the marketplace.
- Identify the competitive pressures that drive market and reusable software component improvements and changes, marketing strategies, etc.
- Verify that reusable software components can be appropriately integrated into the system.

- Establish the veracity of one or more potential solutions that incorporate reusable software components.

Detailed information about useful data to collect for market-watch activities in included in Appendix G, Market Watch.

9.9 COMPONENT TRACKING AND STATUS REVIEW

Describe the interactions between the information contained in the CRSMP and progress tracking and status reviews.

- How does the management of COTS and other reusable software components affect progress tracking and status reviews?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Common component development and management follows the tracking and status review process requirements defined in the SDP. The components' progress and status is reported using the mechanisms defined in the SDP and the SMP.

Component maintenance and support is addressed in the software transition plan.

In addition to the traditional oversight actions, the software management organization exercises appropriate oversight in areas such as

- processes identified in this document and other processes as they relate to reusable software components
- component upgrades and technology refresh issues
- component cost estimation, profiles, and projections
- scalability and ability to evolve of the system architecture

9.10 TRAINING AND SUPPORT

Describe the interactions between the information contained in the CRSMP and training and support.

- How does the management of COTS and other reusable software components affect training and support?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Reusable software components can place additional burden on training and support organizations because users and developers are unfamiliar with the components and the components can be upgraded frequently. Development organizations generate plans that training staff can use for reusable software components within their purview. These plans are documented as part of the SRIDB and address

- staffing and funding for training
- component provider involvement in training, if necessary
- additional training following major releases

Plans are also developed for training personnel to perform following tasks:

- managing reusable software component requirements
- developing reusable software component criteria
- evaluating and selecting reusable software components
- architecting, designing, implementing, integrating, and testing when using reusable software components
- managing risks
- executing other processes as necessary

Development organizations, in conjunction with the reusable software component manager, initiate strategies to support all reusable software components. These strategies implement all requirements as specified in the transition to support plan and are documented in the SRIDB. Topics addressed by the transition to support plan include

- help desk, online help, and other support for developers, testers, and early users
- role responsibility for managing the range of minor problems to critical failures, including escalation of critical problems through the hierarchy
- mechanisms for disseminating information to appropriate parties, including rapid communication in the event of critical failures that potentially affect system safety, security, and viability
- support coordination with the component provider

Development organizations work with maintenance and user organizations to transition skills and responsibilities as appropriate.

9.11 COST ESTIMATION AND MODELING

Describe the interactions between the information contained in the CRSMP and cost modeling.

- How does the management of COTS and other reusable software components affect cost modeling?
- What things are the same? What things are different?
- Are any special roles and responsibilities required?

Sample Text

Cost modeling for COTS and other reusable software components occurs in the context of overall program cost modeling. (Refer to the program management plan.) In addition to many traditional costs, the program identifies any new or changed cost factors for COTS and other reusable software components. These cost factors include

- evaluating new product releases, including impact analyses on other system components
- developing evaluation tools and testbeds
- analyzing and modifying glue code, parameters, and other capabilities used to integrate components with the rest of the system
- performing market-watch activities that evaluate technology refresh opportunities and marketplace changes, including direct communication with component providers, conferences,

and vendor forums.

- tracking annual licensing fees, warranties, and data rights

The program monitors and accounts for these cost differences across the life cycle.

The program selects, develops, or refines a cost-estimation technique appropriate for the situation. There can be multiple cost-estimation techniques, depending on the approach and type of reusable software products used to construct the system and the anticipated service lifetime of the system. The program identifies and collects the appropriate metrics to calibrate and maintain its cost-estimation technique. The program also tracks actual versus estimated costs.

9.12 REUSABLE SOFTWARE COMPONENT PROVIDER RELATIONSHIPS

Describe relationships with reusable software component providers.

- What is your strategy for maintaining strong relationships with component providers?
- Are any special roles and responsibilities required?

Sample Text

One of the best strategies to maximize benefits gained from reusable software components is to develop strong relationships with component providers (e.g., vendors). These relationships provide insight into current and future component releases, encourage cooperative exchange regarding current and future capabilities, and minimize miscommunication.

To develop strong relationships with reuse component vendors/providers, reusable software component managers perform the following tasks:

- Develop a strategy to manage vendor/provider relationships.
- Understand and monitor the vendor's/provider's long-term approach and plans for maintenance and support.
- Engage in meetings and exchanges with the vendor/provider and related groups.

- Establish liaisons with other customers (or potential customers) of the vendor/provider.
- Coordinate government vendor/provider relationships with the program contractor vendor/provider relationships in cases where both exist.
- Encourage and facilitate working relationships among different vendors/providers.

9.13 REUSE COMPONENT METRICS

Describe the metrics collected.

- What metrics are collected?
- How and when are they reported?

Sample Text

Management, in collaboration with the software teams, determines the measurement requirements from the SDP that apply to COTS and other reusable software components. Minimally, we will report the following information on a quarterly basis:

- progress in evaluating and selecting reusable software components
- stability and quality of requirements related to reusable software components
- changes to expectations regarding component use
- summary and analysis of defects in components
- emerging problems, issues, and risks related to the component and component vendors/providers
- expected component release schedules
- summary of adaptation code and data (e.g., tailoring, wrapping, data loads) developed for reuse of a component

9.14 HEALTH CHECK

Describe the system health check.

- What is your strategy for a system health check?
- Are any special roles and responsibilities required?

Sample Text

The status of reusable software components is reviewed on a biannual basis to determine whether existing strategies remain valid. The reusable software component status review is chaired by the reusable software component manager. Attendance is open to representatives of all affected suppliers, software teams, and the engineering leadership team.

The reuse component status review considers four primary sources of information:

1. information gathered by the reusable software component manager while maintaining vendor/provider relationships
2. information summarized from reusable software component metrics
3. information gathered during the market-watch activity
4. information from software teams about evolving expectations for the component

The results of the reusable software component status review are used as appropriate to build or modify plans for component use, initiate further risk reduction and development cycles, initiate new make-or-buy or reuse decision-making processes and component evaluations, and for other purposes identified by software teams and the engineering leadership team.

Appendix A EPIC Overview

Describe the Evolutionary Process for Integrating COTS Based Systems (EPIC).

- Does the development require integration of COTS software components?
- Is a spiral development model right for this effort?

Sample Text

EPIC is an engineering process for systems built using COTS software components [Albert 2002]. Throughout the system life cycle, EPIC

- helps analyze and leverage marketplace forces
- facilitates interaction among stakeholders
- synchronizes development and needed business process changes
- measures progress and mitigates high priority risks

An organization using EPIC simultaneously considers factors from the four spheres that must be orchestrated to achieve a system solution:

1. stakeholder needs and business processes
2. product marketplace
3. system architecture and design
4. programmatic (budget, schedule) and risk considerations

Following EPIC, a project proceeds through multiple iterations with activities designed to converge the four spheres through simultaneous refinement and tradeoffs. The result is a continuous increase in corporate comprehension of, and stakeholder buy-in to, the eventual solution. This progress can be depicted as a wedge with the spheres increasingly overlapping as time goes on.

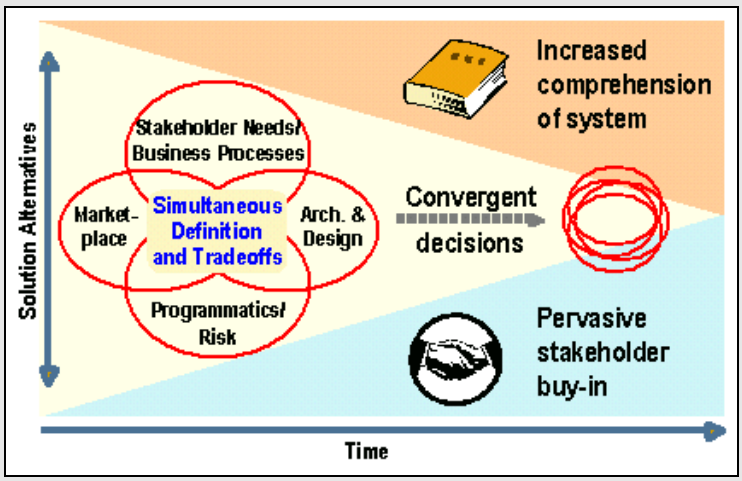

Figure 3: Solution Convergence

EPIC expands upon the Rational Unified Process® (RUP®). As in RUP, the iterations are grouped in four phases demarcated by defined anchor point agreements. For each phase, EPIC provides guidance on

- the area of greatest increase in corporate comprehension
- the activities to be conducted
- the artifacts to be produced

For more information about EPIC, visit the following Web sites:

- EPIC Web page : http://www.sei.cmu.edu/cbs/epic/index.html
- Two Minute Overview: http://www.sei.cmu.edu/cbs/epic/overview.html
- EPIC Overview: http://www/publications/documents/02.reports/02tr009.html
- EPIC Process Description: http://www.sei.cmu.edu/publications/documents/02.reports/02tr005.html
- EPIC Phases and Anchor Points: http://www.sei.cmu.edu/cbs/epic/phases.html
- EPIC Activities: http://www.sei.cmu.edu/cbs/epic/activities.html

® Rational Unified Process and RUP are registered in the U.S. Patent and Trademark Office by IBM Corporation.

- EPIC Artifacts: http://www.sei.cmu.edu/cbs/epic/artifacts.html
- EPIC Executable Representations: http://www.sei.cmu.edu/cbs/epic/model.html

Appendix B OAR Overview

Describe a method for reuse component evaluation.

- Does the development require formal management of reuse components?
- How does one efficiently and cost-effectively rehabilitate legacy software?
- Which legacy components are worth extracting for reuse in a new software product line or with a new architecture?
- What types of changes need to be made to the components?
- What are the risks and costs involved in identifying and reusing legacy components?

Sample Text

The Options Analysis for Reengineering (OAR) method is a systematic, architecture-centric approach for identifying and mining reusable software components within large, complex software systems [Bergey 2001]. The OAR method consists of five major activities with scalable tasks:

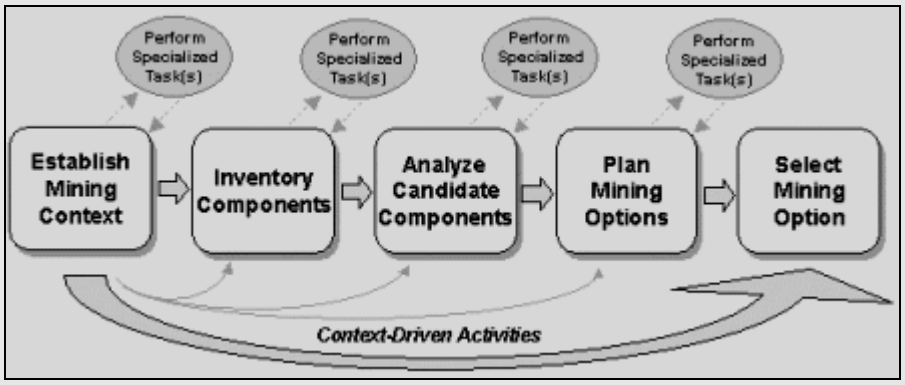

Figure 4: The OAR Process

The five activities have the following goals:

1. Establish Mining Context–Understand the organization's product line needs, legacy base, and expectations for mining legacy components
2. Inventory Components–Identify the legacy system components that can potentially be mined for use as product line components in core asset base
3. Analyze Candidate Components–Analyze a candidate set of legacy components to evaluate their potential for use as product-line components
4. Plan Mining Options–Develop alternative plans for mining based on schedule, cost, effort, risk, and resource considerations
5. Select Mining Option–Select which mining option or combination of options can best satisfy the organization's mining goals by balancing the programmatic and technical considerations

OAR enables users to screen candidate software components; identify the best candidates for reuse; analyze, isolate, and aggregate candidate components; and estimate the level of difficulty, cost, and effort required to mine and rehabilitate the software components selected. Using OAR results, a reengineering team can focus its efforts on those high-value components that meet the technical and programmatic needs of the software product line or the new single-system architecture.

Outputs of OAR include

- inventory of existing legacy components and related documentation
- component Table that identifies reusable components, their characteristics, estimates of changes needed, level of difficulty, estimated effort and cost
- options Table that identifies a set of mining options that reflects the organization's elicited needs, priorities, and concerns
- list of product line or new single system components that can or cannot be satisfied by mining

For additional documentation, see

Bergey, John; O'Brien, Liam; Smith, Dennis. *Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets* (CMU/SEI-2001-TN-013), Pittsburgh, PA: Software Engineer-

ing Institute, Carnegie Mellon University (2001).
http://www.sei.cmu.edu/publications/documents/01.reports/01tn013.html

Bergey, John; O'Brien, Liam; Smith, Dennis. *OAR: Options Analysis for Reengineering: Mining Components for a Product Line or New Software Architecture*, presented at International Conference on Software Engineering, 2001 (ICSE, 2001), Toronto, Canada, May, 2001.
http://www.sei.cmu.edu/reengineering/icse2001/icse2001_1.htm

Appendix C PECA Overview

Describe a specific evaluation process for COTS software components (PECA).

- Does the program require a formal COTS software evaluation and selection process?
- Does this process match the scale of the development effort?
- Who is the COTS software selection decision authority?
- What make COTS software selection difficult?

Sample Text

The growing use of commercial software products in large systems makes evaluation and selection of appropriate products an increasingly essential activity. However, many organizations struggle to select appropriate software products for use in systems. As part of a cooperative effort, the SEI and National Research Council Canada have defined a tailorable COTS software product evaluation process to plan, establish, collect, and analyze (PECA) that can support organizations in making sound product decisions [Comella-Dorda 2003].

Although the PECA process was derived in part from ISO 14598, the process was freely adapted to fit the needs of COTS software product evaluation [ISO 1999]. The process begins with initial planning for an evaluation of a COTS software product and concludes with a recommendation to the decision maker. The decision itself is not considered part of the evaluation process. The aim of the process is to provide all of the information necessary for a decision to be made.

PECA was named for the four main activities that make up the process, which is depicted in Figure 5.

1. Planning the evaluation
2. Establishing the criteria
3. Collecting the data
4. Analyzing the data

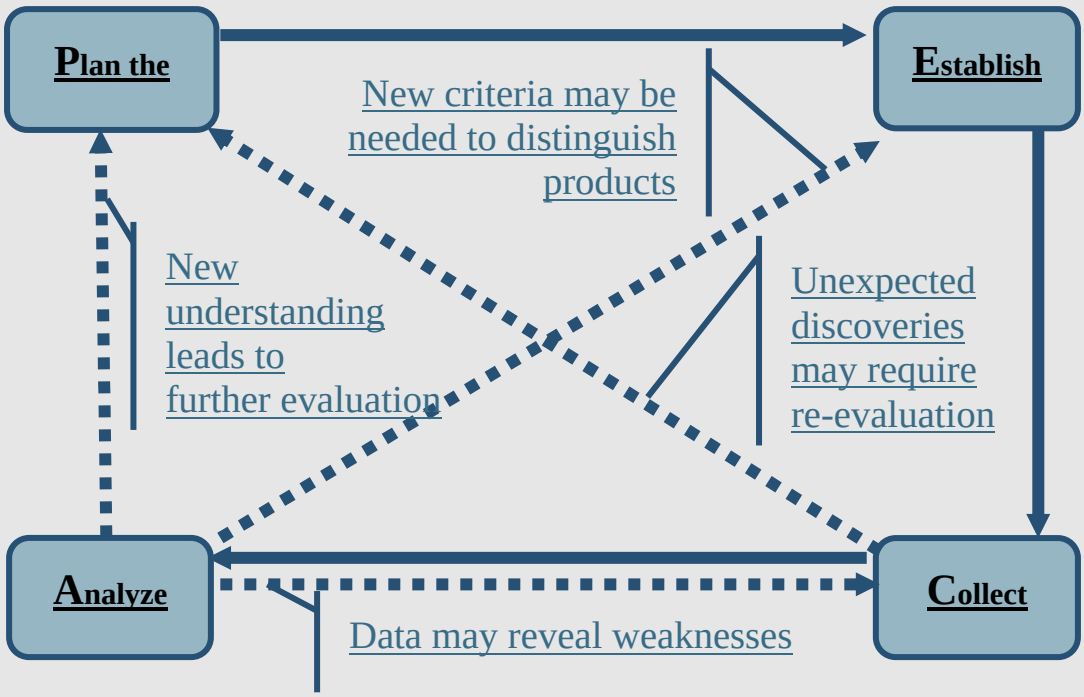

Figure 5: PECA Activities

For additional documentation, see the technical report A Process for COTS Software Product Evaluation [Comella-Dorda 2003].

Appendix D Guidelines for Generating Evaluation Criteria

Sound component evaluation criteria for COTS and other reusable software components.

Note: This sample text provides a taxonomy that can serve as a starting point, but does not provide actual criteria.

- What value equation drives the make-or-buy decision?
- Does this process match the scale of the development effort?
- Who is the make-or-buy decision authority?
- What stakeholders need to be involved?
- Who is responsible for maintaining the criteria?

Sample Text

Component Evaluation Overview

The information contained in this appendix is used for determining whether there are cost and schedule benefits associated with reusing a software component as opposed to building a new one [Albert 2002]. For purposes of this appendix, “software component” refers to collections of one or more software and/or data units most conveniently managed as a unit of capability. The term “use target” means the part of the system in which the software component being evaluated is intended for use.

The COTS and other reusable software components’ evaluations use the guidelines for generation of component evaluation criteria that are described below. These guidelines provide a taxonomy of potential criteria for evaluating a reusable asset. It is important that these criteria be tailored for each specific evaluation. The evaluation of any specific asset will only use the characteristics that are most relevant for that asset.

For example, the functional capabilities of vehicle management subsystem and a C4ISR subsystem will vary substantially. The functional capabilities for different types of components need to be specified. As a second example, Table 2 identifies the evaluation category of "architecture compatibility" and the guidelines of "interfaces are compatible with intended use in the target." The specific interface needs for a component are defined prior to a reuse evaluation.

Table 2: Initial Reuse Screening Guidelines

Evaluation Category	**Guidelines**
Functional Suitability	Good match between product capabilities and system requirements, preferably based on knowledge gained through development or use of the product. Cost to reuse is estimated to be less than cost of new development.
Architecture Compatibility and Standards Compliance	Designed for reuse. Architecture and interfaces are compatible with the system target. Application can be ported to use the system software infrastructure and user interface without significant redesign. Conforms to required standards.
Hardware and Software	Minimum hardware configuration and maximum communication needs are consistent with the system target. Software has been ported to multiple computing platforms. Software footprint and performance are suitable for the system target. Does not depend on products not otherwise planned for use in the system.
Interoperability	Data model/format/access methods are suitable for the system target. APIs are compatible and control mechanisms are compatible.
Quality Factors	Performance, usability, supportability, reliability, safety, and security are consistent with system needs.
Component Provider Acceptability	SW supplier is stable and software to be reused is part of core product line. Will grant Government Purpose Rights.
Product Support Acceptability and Licenses	Product is supported and in use in similar applications. Quality documentation is available. License fees are affordable when compared to alternatives.
Component/System Relationship	Product can be readily configured, adapted (through environment variables and parameters), and otherwise tailored for use in the system.

Table 2, which was derived from the Component Evaluation Taxonomy, lists a set of evaluation categories and guidelines. These categories and guidelines need to be made concrete for specific reuse evaluations.

The Component Evaluation Taxonomy

The Component Evaluation Taxonomy provides a detailed list of potentially desirable component characteristics. Each characteristic is supported by a set of related questions. Items in the list do not represent actual criteria, since they are informally stated.

The items in the Component Evaluation Taxonomy represent a starting point for an evaluation. These items are tailored as appropriate by the Software Team, with input from the supplier, for use in performing a specific evaluation.

The process is summarized below:

1. The Software Team, with input from the supplier, selects those characteristics that are most relevant for a specific evaluation activity. Only those characteristics that are selected in this step are used.
2. The Software Team, with input from the supplier, provides a set of specific criteria to be used for the specific evaluation. The criteria consist of a statement of the expected capability and an approach for measuring whether the component meets expectations. For example, consider the question “Does the component offer appropriate functional capability?” To make this item actionable, the Software Team/supplier would insert the specific type of relevant functionality that is required.
3. If a characteristic is important for a specific evaluation, and is not listed in the taxonomy, that characteristic is defined for the specific evaluation along with the specific criteria to be used for the evaluation. If the characteristic has potential for being applicable to other evaluations, it is added as a change request to the Component Evaluation Taxonomy.
4. After the guidelines are tailored by the Software Team, with input from the supplier, for a specific evaluation activity, each software supplier performs evaluations of software components that are candidates for reuse according to the criteria developed by the Software Team. Results of evaluations are documented in Component Evaluation Records, as shown in Appendix E.
5. When a software component is dependent on another software component, an evaluation of that component should also be reported.

Functional Suitability	
Appropriateness	Does the component offer appropriate functional capability? Is this functionality provided in an appropriate manner (appropriate process, interfaces, quality, etc.)?
Operational Process Consistency	Are the operational processes assumed by the component appropriate to the use target? Which SYSTEM operational processes must change to use the component? How will this change be accomplished?
Completeness	What proportion of the use target capability does the component provide? How was this determined? What is the mismatch between the functions necessary in the target system and those supported by the component? What level of effort will be required to provide missing capabilities or enhance deficient capabilities? How should this be accomplished?
Tailoring/Customization	Is the component suitable “out of the box” or does it require custom construction of scripts, code, tables, and so forth.? What effort is involved in performing this customization? Who will perform this customization? How long will it take? What skills are necessary? Must this effort be repeated to incorporate new component releases? If the use target is part of a system product line and must provide functionality that varies between instances of systems, does the component provide build-time or run-time mechanisms to allow developers or end users to choose the correct functional capabilities?
Variability Mechanisms	What mechanisms exist to change or tailor the functionality or capabilities of components? What is the binding time for these mechanisms and is it appropriate for the system?
Control Flow	Are the methods used by the component for transfer of control to other components in the system (i.e., subroutine call, remote procedure call, data stream, sockets) appropriate for the use target? Is the method used by other components in the system to transfer control (i.e., subroutine call, remote procedure call, data stream, sockets) to this component appropriate for the use target?

Synchronicity	If the components' actions are dependent upon the state of other components in the system (i.e., sequential, parallel, synchronous, asynchronous, opportunistic), is this dependency acceptable for the use target?
Data Flow	Are the mechanisms used by the component to share data with other components (passed, shared, broadcast, copy-out copy-in) appropriate for the use target? Are the frequency and amount of data shared with other components (continuous, sporadic, low volume, high volume) acceptable for the use target?
Binding Time	When is the identity of the component in a transfer of control operation established (i.e., write-time, compile-time, link-time, invocation-time, runtime)? Is this appropriate?
Architecture Compatibility	
Component	Was the component designed to be reused? Are the architectural paradigms employed by the component (i.e., event system, batch sequential, repository, object-oriented) appropriate for the use target? Are the component interfaces compatible with the use target? Has the component's architecture been specified and documented and is it appropriate for the use target? Have the architectural assumptions the component makes about its environment been documented and are these assumptions appropriate for the use target?
Architectural Restrictions	Does the component impose architectural restrictions on the system (topology, standards)? Are these appropriate and acceptable for the system? Is the impact on other system components acceptable for the system?
Architectural Interactions	Is the component architecturally compatible with the components with which it must interact? If the component is not directly compatible, what technologies (e.g., wrappers, bridges) and effort are required to bring it into architectural alignment?

Standards Compliance	
Standards Compliance	Does the component comply with the standards required for the use target? If other standards are supported, do they conflict with standards specified for the use target? If there is no conflict, is it appropriate to add the standard to the list of approved system standards (TV-1)? (Standard should be widely supported and maintained by a standards body.)
Completeness	Does the component implement a subset of the standard, the complete standard, or a superset of the standard? What are the plans for update or enhancement to subsequent versions of the standard?
Confidence	How is standards compliance verified?
Hardware Compatibility	
Configuration	What is the minimal hardware configuration (computers, processors, memory, disk, bus, peripherals, etc.), recommended configuration, and maximum configuration? What incremental steps can be made in hardware to increase performance and storage capacity of the system? Is the required hardware configuration compatible with the use target? Are there any known compatibility problems between the component and expected hardware components?
Communications	If a communications infrastructure is required, is it compatible with the communications infrastructure that will be available to it in the system, including bandwidth?
Security	Is the security of all hardware components within the required configuration appropriate for the use target?
Reliability	Is the reliability of all hardware components within the required configuration appropriate for the use target?
Vendor or Provider Characteristics	Are component provider characteristics for all hardware components within the required configuration appropriate for the use target?
Upgrade	How is the upgrade of a hardware component tied to upgrade of the component? How long after upgrade of hardware is a component upgrade generally available? How long are old versions of hardware supported by the component?

Software Characteristics	
Operating System	Has the software been deployed on the operating system(s) being considered for use on the use target (including versions)? Are the performance and size characteristics appropriate for the use target? Does the component use proprietary operating system features? What mechanisms exist to identify and resolve problems related to the interface between the operating system and components?
Communications	What communications support is required (including versions)? Are alternative communications capabilities supported? Are the performance and size characteristics appropriate for the needs of the target system? What mechanisms identify and resolve problems related to the interface between communications and the component? Who is responsible for identifying and resolving the problem?
Database	What database support is required (including versions)? Are alternative databases supported? Are the performance and size characteristics of the supported database(s) appropriate for the needs of the use target? What mechanisms exist to identify and resolve problems related to the interface between the database and the component? Who is responsible for identifying and resolving such problems?
Accuracy	Is the accuracy of all units with the software configuration appropriate for the use target?
Source Code	Is source code for the product available? Will the product require re-compilation for the use target? Are the language and compiler appropriate for the use target?

Related Components (Dependencies)	What other components are required (including versions)? Are there alternates for these components or can the dependencies be eliminated? Would these components be selected for the use target if they were not required by the component being evaluated? Are the performance and size characteristics appropriate for the needs of the target system? What mechanisms exist to identify and resolve problems related to the interface between the related applications and the component? Note: Component evaluations should be performed for the required components.
Compatibility Problems	Are there any known compatibility problems between the component and any other software component?
Interoperability	
Data Model/Format	What data model and formats does the component employ? Are they published? What standard are they based on? What other components support the same data model/formats? Is there cost and/or schedule impact required to make the data model/formats compatible with the use target?
Support for Data Access	What interfaces or techniques are available to access component data? What effort is required to access component data? Is the granularity of data access appropriate for the target system?

Support for Control	Can the component be invoked by other components? How is it invoked and at what granularity? Can other components control low-level functions that might be necessary in the integrated system (for example, commit for a change)? Can the component invoke other components? How are these invoked? What constraints are placed on these invocations? How can execution of the component and other components be synchronized? What timing concerns may arise? Is there cost and/or schedule impact required to make the control mechanisms compatible with the use target?
Performance	
Responsiveness	What is the response time under light load? Average load? Peak load? Can response times be tuned or improved?
Benchmarking	Are performance benchmarks available for this component? Are the results of these benchmarks suitable? Do the benchmarks reflect a usage situation or pattern consistent with that expected of the component in the target system?
Time-Related Behavior	Does the component exhibit appropriate time-related behavior (throughput, lack of deadlock, thread-safety, latency, etc.)? Is there any potential for time-related interactions with other system components? Where? Have these interactions been evaluated and determined to be within acceptable limits or risk levels?
Resource Behavior	Does the component make appropriate use of resources (processors, memory, devices, etc.)? Is there a possibility of contention for resources with other system components? Have these contentions been evaluated and determined to be within acceptable limits or risk levels?
Surge Capacity	Does the component have the capability to handle increasing loads as expected (e.g., increased number of transactions, increased complexity of processing, increased number of tracks, etc.)?

Adaptability/Flexibility	Can the component be tailored to efficiently handle an appropriate range of performance expectations (transaction rates, numbers of tracks, etc.)? How is this adaptation accomplished?
Human-Machine Interface Usability	
Skill Level Required	What skills do users require?
Responsiveness	What is the response time under light load? Average load? Peak load? Can response times be tuned or improved?
Help Capabilities	What help capabilities are available in the component?
Error Assist/Recovery	How does the component respond to erroneous input and operator error? How does the component assist users when they make an error in input of data? How does the component support users in recovery from erroneous input?
Understandability	Is the component easy to understand? Are common usage paradigms employed?
Learnability	How long will it take before users will be proficient with the component?
Core Capability	Is the component considered to be in the provider's core product line?
Organizational Stability	Has the organization existed in its present form for a suitable period to indicate that it is stable?
Financial Stability	If the organization is commercial, is it making money? What are the financial trends?
Nationality	Are all developers U.S. citizens?
Ease of Access	Is there sufficient access to the organization for answering technical and business questions?
Independence	Does the organization make independent decisions, or is it (effectively) controlled by another organization? Are the goals and directions of the controlling organization appropriate for the needs of the target system?
Reputation	Does the organization have a reputation for quality? Is delivery timely? Is it responsive to customers?

Support Infrastructure	Does the organization offer local offices, hotlines, installation and integration support, and so forth? Does the organization explicitly encourage customer collaboration/sharing via forums such as on-line collaboration bulletin boards and annual User Group meetings?
Engineering Approach	Is the organization's engineering approach (e.g., planning, design, implementation, integration, testing, configuration management, change control, bug tracking,) appropriate for the use that will be made of the component (e.g., proof of concept, prototype, full-scale engineering)? Has the product development organization been certified at SEI SW-CMM®/CMMI-SW Level 3 or above?
Maintenance Approach	How long are old versions of software supported by the component provider? Is the maintenance approach appropriate and compatible? Does the organization provide explicit visibility of the product evolution plans, such as by publishing to customers a list of open bug reports, a list of open enhancement requests, and a schedule for bug fixes/enhancements and new releases? Will the organization continue to be responsible for maintenance and evolution of the component?
Intellectual Property	Will the provider grant government purpose rights for the software component?
Product and Support Acceptability	
Intended Use and Users	Are the intended users and intended use of the component consistent with its intended use for the system?
Availability	Is the component ready for delivery? If not, when is the expected release date? When was the component first made available to customers?
Component Stability	What is the release history of the component? What types of changes were made for various releases?
Installed Base	How many copies of the component are in use? How many uses of the component are similar to the intended use in the system? Has the use of the component by other organizations been verified (i.e., proven or shown not to be marketing hype or shelfware)?

Customer References	What customer references are available? How do these customers use the component, when did they take delivery, how many copies of the component do they use, and how many users are supported? What are their impressions of the component provider, component, support, ease of use, and so forth? Is the use of the component by these customers similar to the anticipated use of the target organization? Have customers of the component used it in a software product line or system? What modifications, wrapping, or translation was needed to make the component compatible with the architecture of other components in product lines or systems?
Training	What training is available for the component, when and where is it offered, and how much does it cost? Is training available for an appropriate set of stakeholders (system personnel, maintainers, end users, and so forth)?
Hotline	During what hours of operation is a hotline available? What types of support are available? Are hotline calls fielded domestically? Are there appropriate capabilities to maintain required security?
Consultants	Are component provider-sanctioned consultants available? Are third-party consultants available? What is the availability and cost for consulting?
Documentation	Is the quality of all documentation and other information appropriate? Is available design information sufficient to determine whether the design is appropriate? Is it sufficient for determining an integration strategy with other target system components? Can materials be reproduced as needed?
Maintenance Information	Is the available maintenance information sufficient for installation? For routine use? For preventive maintenance? For fault isolation and recovery?

Customization	Can documentation, training materials, design information, maintenance information, etc., be customized for unique target system needs? What is involved in customization? What will it cost?
Impact on System Supportability	
End-of-Life Plans	What phase-out or end-of-life planning is the component provider considering? When is phase out or end of life planned? What will the upgrade path be? What will this upgrade require of users? Are any plans documented and available to customers?
Upward Compatibility	Have all versions of the component been upward compatible? Which versions have not been and why? What steps must be taken when a new release of a component must be installed?
Site Installation Support	Who is responsible for installation of the component on site? Will the component provider install the component? Is there extra cost for this service? Can target organization personnel install the component? What skills are required?
Site Operation Support	Will the component provider allocate personnel to support initial operations, perform standard maintenance, or diagnose errors? Does the component indicate to users/operators when maintenance is necessary or an error has occurred?
Analyzability	Does the component provide capabilities to analyze performance? To locate problems or bugs? If capabilities are not provided, how is this accomplished?
Replaceability	If the component must be replaced with another commercial component, what changes would be necessary to the system? What activities would be necessary for data migration?

Preventive Maintenance	Is periodic preventive maintenance required? What activities are involved and how frequently are they to be performed?
Special Support	What equipment, components, and tools are required in the software engineering environment to use the component? Does the vendor provide the functional and non-functional test cases and repeatable test suites/scripts for each product release?
Reliability	
Reliability	Are any claims made about reliability? Is the advertised reliability of all units within the software component appropriate for the use target?
Benchmarking	Are reliability benchmarks available for the component? What is mean time between failures for the component?
Test Regimen	How does the component provider perform testing? Are the results of testing independently verified? Are test scripts and results available?
Type/Frequency of Faults	What types of faults are known and how often do they occur?
Recovery from Faults	What is the error-handling strategy? Is there journaling of faults? Are all faults trapped before the system panics?
Experience	What systems requiring similar reliability to that of the target system use the component?
Safety	
Provider	What is the provider's track record for producing safety-critical components? What is the provider's responsiveness to flaws in safety-critical components?
Visibility	Is the design documentation and source code available for inspection? Is the provider's management and engineering process visible and appropriate?

Test Regimen	How does the component provider perform safety testing? Are the results of testing independently verified? Are test scripts and results available? Have analyses of the types and severity of safety errors been performed? What is the frequency of different sorts of errors?
Certification	Is the component safety certified? By whom? In what context? What safety standards does the component meet?
Benchmarking	Are safety benchmarks available for the component? Are any claims made about safety? How are those claims supported?
Mechanisms	What mechanisms are used to recover from an unsafe state to a safe state? What mechanisms does the system use to protect against flaws in component algorithms? What approaches are used to protect against multiple safety failures due to a common cause or propagation of safety faults? What approaches are used to isolate safety-critical/related parts of the component from non-safety related parts? Does the component detect unsafe states? How does the component respond in unsafe states (e.g., shut down, assume some default state, operate in a safe but degraded mode, reject input that causes unsafe states)
Experience	What systems require similar safety to the target use component?
Security	
Overall Security	Is the security architecture of all units within the software components appropriate for the use target?
Provider	What is the level of trustworthiness of the provider? How is that level determined? What is the provider's history of producing secure components? What is the provider's responsiveness to security flaws?

Visibility	Is the design documentation and source code available for inspection? Is the management and engineering process visible and appropriate?
Test Regimen	How does the component provider perform testing? Are the results of testing independently verified? Are test scripts and results available? Is the provider willing to cooperate with third-party certification or testing? Was the component independently tested?
Certification	Is the component certified? By whom? In what context? What security standards does the component meet?
Benchmarking	Are security benchmarks available for the component? Are any claims made about security? How are those claims supported?
Mechanisms	What methods are used for authentication of users, communications, and so forth? How is confidentiality of data and communication maintained? How is unauthorized access to data prevented? What mechanisms does the component provide for security audit? What mechanisms does the component provide for recovery from insecure states?
Licenses	
Usage/Maintenance	What are the prices and terms of development licenses, run-time licenses, and annual maintenance? Are license terms negotiable? Is site licensing and/or quantity discounting available? Is the total cost of licenses acceptable for the system and with those costs considered, does the component still provide the most cost effective solution?
Transferability of License	Are licenses transferable to other operating units or other agents working on behalf of the system?

Data rights	What data rights are included in the standard license? Are these appropriate for the use target? Must additional data rights be negotiated?
Escrow	Can source code be escrowed? What are the costs and stipulations of that escrow? Is an escrow a reasonable precaution for this system?
Discontinuation	What rights does the target organization have if the component is discontinued?
Expiration	What events occur when a license expires? Is there any notification of impending expiration? Are licenses “time bombed?”
Excess	Does the component offer additional functional capability that will not be used? Should not be used? What impact does this additional capability have on resource needs, safety, performance, and so forth?
Component/System Relationship	
System Configuration	What system configurations is the component part of or workable with?
System Adaptation	What environment variable settings are required? What specific settings are required for networking, memory, processes, peripheral devices, and so forth? What adaptation and settings are required of other components of the system to work with this component?
Integration	What (new) assumptions or expectations does the unique component version make regarding interaction with other components in the environment? What changes must be made to the assumptions made by the rest of the system regarding the behavior of this version? What integration guidelines must be followed and specific integration activities undertaken?

Tailoring/Modification	What tailoring or modification of the component is required to use the component? Was the component provider consulted regarding this tailoring/modification? What was the provider's response? Will tailoring/modification affect the contract in any way (e.g., changes in license fees, changes in maintenance practices or responsibilities)? If the modified version is assumed to become part of the component baseline, what contractual assurance is there that this will actually happen? Who will perform the tailoring/ modification? What scripts, tables, schemas, 4GL code, and so forth are required to use the component? What settings are required for component variables?

Appendix E Component Evaluation Record

Record details of component evaluations.

- Who generates the component evaluation records?
- Where are they stored?
- Should vendors or authors have the opportunity to challenge or review them?

Sample Text

Component Evaluation Record Overview

The results of the evaluation of software components considered for inclusion in the system will be documented in Evaluation Records. An Evaluation Record is an essential item for project tracking, historical data collection, and evaluation of items for improvement. Evaluation records are stored as documented in Section 7.6, Software Reuse Item Database. [Albert 2002]

Template

At a minimum, the evaluation record will contain the information listed below. Supplier format is acceptable.

Identification

- name of product and version that was evaluated
- whether or not the product is recommended for use in the system
- who led the evaluation (name and company)
- when the evaluation was conducted
- contact information for questions and/or clarifications

Target use component

- name of the system component that is the target for reuse
- description of the target use component and its major functions

- source of requirements for the component
- description of the system software production method (such as automatic code generation)
- estimated Source Lines of Code Size (SLOCS) for the system component

Candidate Reuse Product Description

- formal product name and release or version
- name of the owner (organization) of the Product that contains the candidate reuse component(s)
- names of the candidate reuse components
- reason for focusing on a product subset, if the whole product was not evaluated
- identification of owner proprietary rights, if any
- identification of restrictions for system reuse, if any
- identification of licensing costs, if any
- name and contact information for the person who has the authority to release the product source code and its documentation for system reuse
- brief description of the environment in which it was designed for use

Candidate Reuse Components Information

- provide the following information for each candidate reuse component
- candidate reuse component name
- description of the candidate reuse component
- name of the expert reuse assessor
- assessor credentials. Include system role, candidate reuse component development role, and assessment experience.
- reuse assessor contact information—phone number and e-mail address
- method used for component evaluation (development, product use in similar systems, test, product analysis, review of third-party documents, review of vendor documents, etc.)
- any relevant features or circumstances that could affect the results
- name of the target system component major function
- reuse candidate component source language
- assessment of how well the candidate meets system needs

Estimated Effort to Reuse Components

- reuse candidate component SLOCS
- estimated range (Low, Middle, High)-(10-50-90) of SLOCS required for integration of the candidate component into the system. The (Low, Middle, High)-(10-50-90) range is a measure of uncertainty of the integration SLOCS. It means that there is
  - a 10 percent chance that the actual integration SLOCS will be less than the Low estimate,
  - a 10 percent chance that the actual integration SLOCS will be greater than the high estimate, and
  - a 50 percent chance that the actual integration SLOCS will be greater than or less than the middle estimate

If the candidate component requires modification, provide the following additional information.

- description of modifications required for use in the system
- description of the software production method used to develop the candidate reuse component
- description of the basis of reuse estimate; for example, similarity between reuse component and target component of requirements, design, and execution environment
- names of the component development team
- indication of the members of the development team, if any, that will modify the component for system reuse
- relevant Candidate Component SLOCS
  (If only a portion of the candidate component will be reused estimate the size of the portion, otherwise use the reuse candidate component SLOCS.)
- estimate of (Low, Middle, High)-(10-50-90) range of candidate component reuse SLOCS
- rationale for the low estimate of candidate reuse SLOCS
- rationale for the high estimate of candidate reuse SLOCS
- formula for the cost of reuse
- estimate of the (Low, Middle, High)-(10-50-90) range of cost of new development as opposed to modification of the reuse component

Component Evaluation Record

Construct a table containing the specific evaluation criteria defined according to the process defined in Appendix D. An example is shown in Table 3.

Table 3: Evaluation Criteria

Requirement/ Negotiability	Weight	Capability Question	Measurement Criteria	Evaluation Result

Summary

- component is a viable candidate for reuse in the use target: ___Yes ___No
- limitations or deficiencies
- estimated cost of reuse vs. estimated cost to build new
- other relevant information

Appendix F Reuse Evaluation Analysis Report

Provide comparison mechanism between CERs.

- Who generates/maintains?
- Level of automation?
- Database versus report generation?
- Security issues due to proprietary information between vendors?

Sample Text

The reuse evaluation analysis report summarizes critical requirements and the weighting of the criteria that led to the selection of a component. Information to be included in the analysis report is listed below.

- summary of the evaluation process
  - components considered
- observed features
  - performance against criteria
  - other observed behaviors and interactions
  - location of evaluation record
- comparison of alternatives and identification of selection made
- sensitivity analysis to identify whether selection could stand up under potential changes to requirements and requirement weighting
- for the selected component
  - summary of strengths, weaknesses, and mismatches between component capabilities and expectations
  - proposed resolution of remaining mismatches
  - description of needed
    - tailoring
    - extensions

    - modification
    - porting
    - wrapping
    - data filtering and extraction
    - any other aspect necessary for the component to work within the system context
- description of the interoperability of the component with rest of the system in terms of
  - assumptions
  - capabilities
  - constraints and limitations
- waivers needed for standards
- cost and schedule required and basis for estimates
- identification and prioritization of risks and risk-reduction activities
- limitations or conditions placed on use
- implications for
  - overall system capability and performance
  - overall system level of service for security
  - safety
  - project plan
  - system build increments and order
  - risk management (risks identified, priority assigned, risk reduction activities planned)
  - design
  - integration and test
  - formal qualification testing
  - system infrastructure
  - deployment
  - license management
  - system and subsystem architecture
  - configuration management
  - release management
  - training
- deficiencies in assessment methods
- overall confidence in assessment methods

Appendix G Information Needed for Market Watch

Proactive surveillance of each significant component's market space.

- Who does the market watch?
- Who does budget allocation?
- Who does user group participation?
- Who does contingency planning?

Sample Text

Competitive Market Forces

Identify

- the level of market competition and the number of potential sources capable of satisfying the stakeholder needs
- status, size, and location of sellers in the market
- market prices and pricing trends
- customary terms and conditions governing commercial sales of the component
- factors that affect market prices
- market information on dollar and unit amounts of component sales
- trends in buying practices
- business practices peculiar to the marketplace
- trends in commercial and government sales
- barriers to new firms entering the marketplace
- business growth, expansions, and declines

Buyers, procurement practices, and approaches

Identify

- active component buyers and users
- characteristics (size, function, nationality, etc.) of active buyers and users of components in the market
- business processes supported by buyers and users in the market
- similarity and differences between these business processes and those required
- market technologies and components employed by buyers and users
- related technologies employed by buyers and users
- average dollars spent and other resources expended
- evidence of success or failure of systems incorporating market components
- potential market growth and size
- growth forecasts
- barriers to market growth
- common procurement strategies for buyers and users comparable to this organization
- common implementation strategies of comparable buyers and users
- implementation status for comparable buyers and users

Applicable Industrywide Laws or Regulations

Identify

- pending legislation or regulation
- antitrust or competitive practice litigation
- reports regarding safety problems and fraud in the marketplace

Market Consolidation

Identify buyouts and mergers.

Market Standardization

Identify

- standardization of functional capabilities, technologies, and practices
- bodies governing vendor practices and component capabilities
- consortia and other organizations dedicated to collaboration between component developers and users
- cooperative agreements

Technological Changes and Trends

Identify

- basic technologies competing in the marketplace (e.g., object technology, relational database technology)
- relative maturity of basic technologies
- trends in marketplace acceptance of common technologies
- components representing competing technologies
- emerging technologies
- relative maturity of emerging technologies
- market penetration for emerging technologies
- projected market penetration for emerging technologies

Component Changes and Trends

Identify

- new components in development for future availability
- technologies employed in new components
- functional changes expected in new components
- cost/complexity of upgrade from current to future components

Appendix H Acronyms and Abbreviations

COTS
commercial off-the-shelf

CRSMP
COTS and Reusable Software Management Plan

GOTS
government off-the-shelf

Appendix I Glossary

Application Programming Interface (API)
An API specifies what a software designer needs to know in order to use a defined set of software services. The API defines both what the services are and how they can be accessed in a program.

Can Make or Can Buy
Software work effort which cannot be categorized as either must make or must buy. An option to make or buy exists, based on the evaluation criteria of performance, quality, cost and schedule.

COTS (Commercial Off-the-Shelf) Product
Any software, hardware, or service item that is offered for sale, lease, license, or free of charge to the general public in multiple, identical copies and used without modification of the internals. COTS products are supported and evolved by the vendor, who retains the intellectual property rights.

References

URLs are valid as of the publication date of this document.

[Albert 2002]
Albert, Cecilia & Brownsword, Lisa. *Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview* (CMU/SEI-2002-TR-009, ADA405844). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.
http://www.sei.cmu.edu/publications/documents/02.reports/02tr009.html

[Bergey 2001]
Bergey, John; O'Brien, Liam; & Smith, Dennis. *Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets* (CMU/SEI-2001-TN-013, ADA395201). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001.
http://www.sei.cmu.edu/publications/documents/01.reports/01tn013.html

[Comella-Dorda 2003]
Comella-Dorda, Santiago; Dean, John; Lewis, Grace; Morris, Edwin; Oberndorf, Patricia; & Harper, Erin. *A Process for COTS Software Product Evaluation* (CMU/SEI-2003-TR-017, ADA443491). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
http://www.sei.cmu.edu/publications/documents/03.reports/03tr017.html

[ISO 1999]
International Organization for Standardization (ISO). *ISO/IEC 14598-1:1999–Information Technology–Software Product Evaluation.* Geneva, Switzerland: ISO/IEC, 1999.

[Saaty 1980]
Saaty, Thomas L. *The Analytic Hierarchy Process*. New York, NY: McGraw Hill, 1980.

[SEI 2007]
Software Engineering Institute, *Software Product Lines*.
http://www.sei.cmu.edu/productlines/index.htm (2007).

[USC 2002]
WinWin Spiral Model, University of Southern California, 2002.
http://sunset.usc.edu/research/WINWIN/winwinspiral.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2007	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
COTS and Reusable Software Management Planning: A Template for Life-Cycle Management	FA8721-05-C-0003

6. AUTHOR(S)

William Anderson, Ed Morris, Dennis Smith, Mary Catherine Ward

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2007-TR-011

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2007-011

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The acquisition community needs guidance in long-term management planning for selecting, approving, and upgrading software products, especially commercial off-the-shelf (COTS) and other reusable software products. As the mixture of these components in systems increases, the demand for a planned way to manage them continues to grow.

The COTS and Reusable Software Management Plan (CRSMP) can facilitate acquisition programs' management of COTS and other reusable software products. The CRSMP provides a strategy outline for managing data about component licensing, tracking release schedules, monitoring software interdependencies, choosing specific features and extensions and documenting those choices, and evaluating and mitigating risks associated with deploying COTS and other reusable software components in a system.

The CRSMP presented in this report can serve as a guide for how to manage multiple COTS and other reusable software components in complex systems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
COTS, acquisition	155

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL